**NAVAL POSTGRADUATE SCHOOL**
**Monterey, California**

**THESIS**

# XML AS A DATA EXCHANGE MEDIUM FOR DoD LEGACY DATABASES

by

Kris Pradeep

June 2002

| | |
|---|---|
| Thesis Advisor: | Valdis Berzins |
| Second Reader: | Paul E. Young |

| REPORT DOCUMENTATION PAGE | | | *Form Approved OMB No. 0704-0188* |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY *(Leave blank)*** | **2. REPORT DATE** June 2002 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** **XML as a data exchange medium for DoD legacy databases** | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Pradeep, Kris | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | | | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited. | | | **12b. DISTRIBUTION CODE** |
| ABSTRACT (maximum 200 words) This thesis addresses the issue of interoperability in DoD legacy system databases and evaluates XML as a tool for transferring message data between varied systems. With the demands for increased communication, the dire requirement for a common mode of information transfer is greatly realized. Many legacy systems have developed their own unique interfaces. XML is one solution which can help ease the transition to a common interface. This thesis is a part of a larger team effort. In contributing to this larger effort, a software program was developed to generate select messages in their native and XML formats. | | | |
| **14. SUBJECT TERMS** Interoperability, Interconnectivity, Legacy Database Systems, XML | | | **15. NUMBER OF PAGES** 111 |
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFI- CATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UL |

**NSN 7540-01-280-5500** **Standard Form 298 (Rev. 2-89)**
**Prescribed by ANSI Std. 239-18**

THIS PAGE INTENTIONALLY LEFT BLANK

# XML AS A DATA EXCHANGE MEDIUM FOR DoD LEGACY DATABASES

Kris Pradeep
B.E., University of Madras, India
M.Tech., Indian Institute of Technology - Madras, India

Submitted in partial fulfillment of the requirements for the degree of

**MASTER OF SCIENCE IN SOFTWARE ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2002**

| | |
|---|---|
| Author: | Kris Pradeep |
| Approved by: | Valdis Berzins, Thesis Advisor |
| | Paul E. Young, Second Reader |
| | Luqi, Chairman, Software Engineering |

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

This thesis addresses the issue of interoperability in DoD legacy system databases and evaluates XML as a tool for transferring message data between varied systems.

With the demands for increased communication, the dire requirement for a common mode of information transfer is greatly realized. Many legacy systems have developed their own unique interfaces.  XML is one solution which can help ease the transition to a common interface.

This thesis is a part of a larger team effort.  In contributing to this larger effort, a software program was developed to generate select messages in their native and XML formats.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

## ACKNOWLEDGMENTS

The author wishes to express his thanks to Dr Valdis Berzins and CAPT Paul Young for their guidance and support.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. BACKGROUND INFORMATION

In this new millennia information is the quintessential element of success in all aspects. This is especially true in the defense industry when the safety of a nation is at stake. It is also equally applicable during maritime operations. As the US and our allies participate in an increasing number of joint military operations around the world, interoperability of military systems becomes absolutely essential. Transfer of data, in a timely fashion, between military computing systems becomes crucial to establishing interoperability.

A large number of real-time systems are in use by today's military in the crucial areas of Command, Control, Communications, Computers, and Intelligence (C4I). All of these C4I systems have legacy databases. The only way to create an interoperable system between them is to develop a methodology to facilitate transfer of information.

Several students at the Naval Postgraduate School (NPS) and the Joint Battle Center (JBC) organization are involved in a major effort to evaluate the interoperability issues in Department of Defense (DoD) legacy system databases. One aspect of this program is the evaluation of XML as a tool for transferring message data between varied systems. This thesis supports this effort by evaluating selected databases and messages and generating the messages in the right format for further evaluation.

## B. SCOPE OF THIS THESIS

The primary effort of this thesis will be to develop programs that generate random data sets for further data manipulation by the other team members. The thesis effort concentrates on four legacy databases and determines the data subsets to be used which would constitute a representative subset of the entire system. The data subsets chosen should support typical interactions among the different databases that might be encountered in system interoperation. The effort also includes looking into the various message formats used for transfer of messages as applicable to the four legacy databases used in the evaluation.

## C. METHODOLOGY

The thesis effort will evaluate four legacy databases and will determine the subset to be used. In a normal system operation, information from these databases are searched and transmitted to other systems. Hence, the messaging formats employed for message transmission will be analyzed. The subset schemas will be further populated with the message formatting rules and parameter ranges. Finally, programs will be written to create messages from the subset containing random (valid) data for transmission.

## D. ORGANIZATION

This thesis is organized in the following manner:

- Chapter I is an introduction to the thesis providing a brief background.
- Chapter II deals with “Interoperability” issues.
- Chapter III discusses the four legacy databases.

- Chapter IV looks into the message formats and also discusses the applicability of XML.
- Chapter V details the selection of message subsets.
- Chapter VI contains details of the XML wrapper used for message transmission.
- Chapter VII details the application of software engineering principles to development of the software.
- Chapter VIII contains the conclusion and also recommendations for further research.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. INTEROPERABILITY ISSUES

## A. INTEROPERABILITY BACKGROUND

The issues of interoperability were first created unknowingly along with the first databases developed for the DOD legacy systems. These very first databases were in fact conceived in total isolation from other systems and hence had no motivation to develop their bases with consideration of interaction with other systems. As time went by, and a multitude of systems came into existence, users began to feel the necessity of accessing information from other databases. Today, command and control has become a thriving industry generating thousands of individual information systems in the United States and across the globe.

## B. THE NEED FOR INFORMATION INTEROPERABILITY

With the advent of the Internet, and the demands for increased communication and data transfer for near real time operation, the dire requirement for a common mode of information transfer was greatly realized. The challenge of interoperability has grown tremendously with the number of systems.

One of the most challenging aspects of this problem is information interoperability, or the ability to exchange information. To be effective in today's military environment, a wide variety of dissimilar information systems, developed independently by different countries must share timely, accurate, complete information in a form that is mutually understandable. They must do so within the competing force's decision cycle despite differences in cooperating force cultures, command structures, operational procedures and languages [MC98]. With the conception of Network-Centric Warfare, the need for Command, Control, Communication, Computer and Intelligence (C4I) system interoperability has become even more critical to the success of military operations.

## C. PROBABLE SOLUTIONS TO INTEROPERABILITY

The systems currently used in the DOD cannot be discarded and replaced with newer systems [RS96]. Their daily use in the current state is very critical to C4I operation. A simultaneous changeover of all systems to any new system is not practical, due to the high risk of failure.

Many legacy systems have developed their own applications which utilize point-to-point interfaces to the other connected systems. This method can become prohibitively expensive, both in terms of development and maintenance, as the number of interfaces increase.

Some other systems have taken the approach of collecting the data from a number of systems and displaying it in “layers” to the user. The problem with this methodology is there is no true integration of data.

Recently, a new technology called the *Extensible Mark-up Language* (XML) has been developed. This technology is being used in the publishing industry extensively. It is also used to a great extent in the E-commerce field to allow interoperability between varied databases in a near-real-time manner. This technology shows great promise for use in the DoD.

## D. RELEVANCE TO THE THESIS

The primary goal of this thesis is to concentrate on selected subsets of four legacy databases and to generate sample messages from these databases for use by the other team members. A prime requirement to construct the program was the need for a transport media to express information independent of the operating platform without losing the metadata or meaning of that information [YBG02]. The Extensible Markup Language, XML, meets these requirements.

THIS PAGE LEFT INTENTIALLY BLANK

# III. DATABASES OF INTEREST

## A. INTRODUCTION

The initial scope of work involved evaluating the following four legacy databases: Joint Common Database (JCDB), Advanced Field Artillery Tactical Data System (AFATDS), Global Command and Control System – Integrated Imagery and Intelligence – Intelligence Shared Data Server (GCCS-I3 ISDS), and GCCS Track Database Manager (TDBM) to determine a suitable subset of message definitions for random generation. Following is a summary of the four databases of interest.

## B. AFATDS

The Advanced Field Artillery Tactical Data System (AFATDS) is a member of the Army Battle Control System. It is the Command and Control system for all ground fire-support systems within the Army and the Marine Corps. The support systems include field artillery, attack helicopters, naval gunfire, mortars, and close air support. Our allies also interface with the AFATDS with a format specifically designed for this purpose. AFATDS can also send and receive a few selected United States Message Text Format (USMTF) messages.

AFATDS is based on Interbase. The information held in AFATDS is not easily disseminated and hence could not be obtained for analysis and evaluation purposes.

## C. JCDB

The Joint Common Database (JCDB) is a key component of the US Army's approach to the use of standard and common software products across all Command and Control systems. JCDB is actually a suite of products designed to efficiently and effectively store and distribute Command and Control data. In the future JCDB will reside on all Army Battlefield Command Systems (ABCS) and will be the information source for a layer of common applications. In order to ease the transition to the JCDB for legacy systems, Application Program Interfaces (API) are being written to act as a translation routine.

The physical database of all ABCS shared data is also called JCDB and is of interest here. This database has been generated in both the Defense Information Infrastructure Common Operating Environment (DII COE) Informix and Oracle systems. The database is currently very large, comprising 528 tables and still growing. In pursuance of this thesis effort information on the JCDB entity/table data and physical naming conventions was able to be obtained. However, an unclassified sample data load from the JCDB could not be obtained in a timely manner. The data sample was essential in determining whether the other team members would be able to use the messages generated from the program to be written. Hence, any further effort to use this database was not pursued.

## D. GCCS

The Global Command and Control System (GCCS) is an Automated Information System (AIS) supporting the DoD. GCCS is the nation's premier system for the command and control of joint and coalition forces. It incorporates the force planning and readiness assessment applications required by battlefield commanders to effectively plan and execute military operations. GCCS is intended to help planners in their crisis planning responsibilities by providing access to a useful, user-tested, integrated set of tools and data. Its Common Operational Picture (COP) correlates and fuses data from multiple sensors and intelligence sources to provide war fighters the situational awareness needed to be able to act and react decisively. Incorporated into GCCS is a large database. All software within GCCS is packaged in modules called software segments. Of the several databases in the GCCS are two databases of interest: GCCS-I3 ISDS and GCCS TDBM, which will be detailed below.

## E. GCCS-I3 ISDS

The GCCS Integrated Imagery and Intelligence (GCCS-I3) enhances the commander's situation awareness by providing ready access to imagery and intelligence information. The GCCS-I3 gives users direct access to the immense global Modernized Integrated Database (MIDB) and the ability to integrate locally obtained imagery and intelligence with national information.

The Intelligence Shared Data Server (ISDS) is one of the many servers that form part of the GCCS-I3 system. The ISDS acts as an access point for imagery and intelligence data from the MIDB.

A detailed evaluation of this database is provided in the thesis submission of Rex Hina [RH99], another team member. The reasons for not recommending to use the GCCS-I3 ISDS as a test bed for XML transformation have been cited in that thesis. One concern with the MIDB is the use of "tie" tables throughout the database for establishing relationships between tables. This has the effect of not only increasing the number of joins required in order to obtain meaningful results, but also can make the transition to XML format more complex [RH99].

## F. GCCS TDBM

The GCCS Track Database Manager (TDBM) is an application tool that manages track information. This database is currently renamed as Track Management System / Universal Comms Processor (GCCS-TMS/UCP). But for all practical purposes it is still referred to as TDBM. TDBM is a segment of the Unified Build (UB). It is one of the many suites available in the GCCS. The capability of the Track Database Management system and the Common Operational Picture is being improved and migrated to Government designated platforms. Messages from this database have been selected for analysis and generation. Messages generated from the GCCS follow a Common Operational Picture (COP) definition and will be discussed further.

## G. GCCS-COP

The GCCS Common Operational Picture (GCCS-COP) provides a number of communication mechanisms for both transmission and reception of data. This is a well established means of disseminating data between GCCS-COP suites and other systems. The capabilities of the GCCS-COP, and the type of data it maintains, has grown beyond the original intended capabilities. Furthermore, extension of the GCCS-COP in the area of interoperability with non-GCCS systems is limited.

## H. THE NEED FOR A NEW MESSAGING STANDARD

The inability of the GCCS-COP to extend the current communications capabilities to encompass new features and data types as well as to enhance systems interoperability has heightened the need for a new messaging standard. These communication enhancements should be text-based to take advantage of the ability to sanitize text-based messages for sharing between Coalition Partners and U.S. systems.  Furthermore, a capability to exchange messages using the Extensible Markup Language (XML) would be greatly beneficial based on the growing international interest in this area.

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. MESSAGE TEXT FORMATS

## A. BACKGROUND

With the proliferation of legacy databases in the DoD, the United States and its allies realized the oncoming Command and Control interoperability problem. They have invested a great deal of time and resources to formalize information standards in order to reduce the ambiguity of natural language and increase opportunities for automation. Over the years, past and present, several formats have been developed and extensively utilized for message transmission. Most notable amongst them are:

USMTF: United States Message Text Format

OTH-G: Over The Horizon Targeting – Gold

AdaTP3: Allied Data Publication, Number 3

CIX: Coalition Information Exchange

TADIL: Tactical Digital Information Links

Of the above, USMTF and CIX will be discussed further.

As discussed in the previous chapter, there exists a need for a common standard for the transfer of messages. The eXtensible Markup Language, XML, seems to foot the bill and will be detailed here also.

## B. INTRODUCTION TO XML

XML is a markup language for documents containing structured information. In order to appreciate XML, it is important to understand why it was created. XML was created so that richly structured documents could be used over the web. The only viable alternative is HyperText Markup Language (HTML) based on Standard Generalized Markup Language (SGML), developed about 20 years ago.

SGML is not a practical choice because of its sheer size. HTML allows an internet browser to display information to users by providing a method to encapsulate text with a fixed set of tags. This predefined set of tags makes the document difficult to read in the absence of a browser. Further, the ability to define new tags is absent in HTML. Hence, HTML is not the best practical choice.

XML documents are human-legible and reasonably clear. Information in an XML document is stored in plain text. Even without an XML browser it can be easily read in any text editor and determine what the content means.

The World Wide Web Consortium (W3C) calls XML "a common syntax for expressing structure in data”. Structured data is information that is tagged to reflect the meaning of its content. For example, whereas the <H1> tag in HTML specifies text to be presented in a certain typeface and weight, an XML tag would explicitly identify the kind of information: <PRICE> tags could contain an item's cost in an inventory list. By not predefining any tags, W3C allows developers full control over customizing their data as they see fit.

XML provides a means for creating common specifications for data shared between systems. By separating structure and content from presentation, the same XML source document can be written once, then displayed in a variety of ways: on a computer monitor, within a cellular-phone display, translated into voice, and so forth. It will work on any communications devices that might be developed.

All of the advantages of XML outlined above make interoperability possible [XML1,XML2]. This helps enable disparate legacy systems share information easily.

## C. USMTF

The United States Message Text Format (MTF) standard is the most prolific of all message formats. USMTF is a text oriented message format. It supports the full spectrum of U.S. military operations, including intelligence, air operations, fire support, maritime operations, logistics, medical, etc.

In the USMTF data standard there are over 350 message types. Each message type is broken into groups of sets known as segments and individual sets. Each set is further separated into fields. The rules about which messages have which sets, and what data must be in each field of a set is contained in the MIL-STD-6040 [MIL1]. Further, the USMTF is designed to support generation of messages that are machine and human readable. Given below is a sample Weather Observation (WXOBS) and a Target Position USMTF message (the headers and other info are removed for clarity).

```
EXER/OLIVE DRAB 93//
MSGID/WXOBS/125TH INF DIV//
WEATHLOC/32WDL123123//
OBSTIME/17090OZ//
CLDLYR/MINCEL:60/CLD:5/CLOUDTYP:CI/80//
VISBY/lOKM/LIGHTRAIN//
TEMP/MAXTEMP:30/MINTEMP:20/FRZLVL:MlOFT//
WIND/010/V:045/15/G:20//
ALTSTG/2992//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  /  20/     25/   325/  25
/MlQKFT /  10/     15/   320/  35//


TARPOS/32WDL123123/141325Z/NNE/25KPH//
TARPOS/3510N07901W/141325Z/025T/MED/130//
```

### 1. Problems using USMTF

USMTF data cannot be uniformly exchanged betwen C4I systems. A specific system may read data from some messages, but no generic method is widely implemented that would easily allow a C4I system to read and parse any USMTF message.

It takes a significant amount of time to reprogram C4I parsing applications in response to changes in the baseline.

Although USMTF messages are designed to be man-readable, they cannot easily be read. The reader requires an in-depth knowledge of the message type sent in order to understand the context of the presented data.

USMTF messages traditionally make extensive use of references; often these references are not available, making it difficult to understand or appreciate the impact of presented information.

The format of USMTF is well established; but its fixed field format wastes bandwidth by sending empty information.

Typically, USMTF provides more information than the destination system requires.

It is not easy to prepare USMTF messages and they are prone to formatting errors.

### 2. A Case for using XML-MTF

The US DoD investigated the possibility of using XML as a message transfer medium or wrapper. In 1998, the MITRE Corporation, supporting the US DoD, began examining XML's applicability to military information exchange. It was soon determined that military data exchange standards, like USMTF, could be very easily expressed in an XML format.

The need was identified early to have a standard way of expressing MTF information in XML format. The name for this new alternate format was coined XML-MTF. This format has been made of the MIL-STD-6040.

Benefits of adopting this XML-MTF standard include [XML3]:

- The first and most immediate benefit of XML-MTF is the flexibility it gives users to display data. Battle Commanders using this standard can display data in any format irrespective of the system used.

- Industry has been developing numerous COTS tools necessary to support XML-MTF data. This will result in increased performance at a lower cost.

- The transmission protocol used by XML-MTF is the same as the World Wide Web (WWW). This means that XML formatted documents can be transmitted over the networks as easily as HTML.

- With this technology, data from one database can be distributed to another different database that holds the same data in a different view easily. Data could be held and retrieved consistently. Several separate databases can keep the same data “picture” in an efficient manner.

- Translation applications between USMTF and XML-MTF format can be easily written.

- Greater security control for classified messages is possible since XML markup can be used to separately classify specific fields rather than classifying the entire message at the highest level of contained information as in USMTF.

- In today’s rapidly changing environment, changes to the USMTF Baseline take a long time to be implemented in the various C4I parsing applications. By using XML-MTF and designing C4I systems to read the well-formatted data, the level of effort required to change systems in respose to baseline updates can be reduced.

- It is possible for XML-MTF to progress to handle message traffic data between USMTF data and Operational Specification for Over-the-Horizon Targeting Gold (OS-OTG), Allied Data Publication Number 3 (AdatP-3), or Variable Message Format (VMF) thereby achieving higher levels of information interoperability within legacy systems.

## D. CIX MESSAGE FORMAT

The GCCS-COP is the host for a vast database of Track information. This Track information was traditionally transmitted and received using the OTH-G messaging format. With time newer data structures have evolved and many of them, like Link Tracks and Theater Ballistic Missile (TBM) Tracks, do not lend themselves to transmission using any existing OTH-G message set due to the nature of their data structure. Clearly the capabilities of the GCCS-COP, and the type of data it maintains, have grown beyond the original capabilities of the OTH-G format. Furthermore, extension of the GCCS-COP in the area of interoperability with non-GCCS systems is limited by the OTH-G format. This requirement necessitated the development of a new format called the Coalition Information Exchange (CIX).

Efforts are underway in the DOD to provide the CIX software to send and receive messages in XML format. This is similar to the XML-MTF program. The XML messaging to be employed by the CIX software has the same outer wrapper as do other GCCS-COP messages, in order to interact properly with the established communications architecture.

# V. SUBSET DETERMINATION

## A. PARAMETERS SELECTED

The main challenge until now was to understand the four legacy databases of interest to JBC and decide what would be an appropriate representation. Due to the difficulties encountered in obtaining real data from the four databases, (for reasons ranging from security issues to unavailability of data), the GCCS-TDBM was chosen as a representative database.

Several message text formats were evaluated, and the USMTF and CIX selected as the formats for message generation. The next step is to select the messages from the database which would be generated in its native and parsed XML modes.

## B. MESSAGE MAPPING

In the development of the XML-MTF Program, a large effort was placed on the XML-MTF mapping. The purpose of this mapping is to set standards for representing the meta-data rich MTF messaging information while making it compatible with COTS supported data formats. In short, XML-MTF mapping describes how XML-MTF messages relate to their respective MTF messages.

## C. MAPPING STRATEGIES

The XML-MTF specification identifies several mapping strategies [XML1]. Some of these strategies were used to select messages from the GCCS-TDBM database for further analysis. A listing of the mapping strategies is given here:

Element Name Mapping

Data Mapping

Elemental Mapping

Composite Field mapping

Alternative Content Field Mapping

Free Text Field Mapping

Empty Field Mapping

Group of Fields Mapping

Linear Set Mapping

Columnar Set Mapping

Amplification and Narrative Set Mapping

Remarks Set Mapping

General Text Set Mapping

Segment Mapping

Message Mapping

Not all strategies were used to select the subset of messages for program generation. An explanation of the mapping strategies used is detailed in sections VI.C.1 through VI.C.4. The goal of this effort is not to go into details of all the mapping strategies with examples; but the goal is to select appropriate messages, which will be representative of the typical interactions that might be encountered in system operation.

## D. MESSAGES SELECTED

The following message sets from the GCCS-TDBM database were selected for generation in its native and parsed XML versions.

- MSGID: Message Identification Message Set
- LCTC: Basic Link Track Message Set
- XPOS: Basic Link Track Message Set
- LEXT: Extended Link Track Message Set

- BMISL: Theater Ballistic Missile Track Message Set
- WXOBS: Weather Observation

The specifications for these message sets are described in detail in the Appendices. In the next chapter the message structures and mapping strategies are detailed.

THIS PAGE INTENTIONALLY LEFT BLANK

# VI. SUBSET SELECTION DETAILED

## A. INTRODUCTION

In the previous chapter, the message sets which will be used for generation of XML formatted messages were selected. In this chapter, the CIX wrapper for these message sets and the mapping strategies used will be highlighted.

## B. XML-CIX MESSAGE DESCRIPTION

### 1. CIX Message Construction

A CIX message is constructed from several components, known as sets. Each set is comprised of a number of fields, some of which are mandatory, and the rest of which are optional [IRI00]. Every CIX message contains a Message Identification (MSGID) set. Following the MSGID set are lines containing the actual message data. The MSGID set is described in Appendix A. An example of a complete message is given below:

```
BT
MSGID/CTG 81.0/CIX/0001/MAY
LCTC/JT4012345678/F14/L11/COMAIRSOUTH
STANDING/1232/AIR/00/FRD/7/132
//1212/2323
XPOS/01211534.5Z2/AUG95/LL:304055.5N2-1304055.5E3/RADAR/150.7T
/123.5NM/76.8NM/120.6T/23.5KTS//23FH
ENDAT
BT
NNNN
```

### 2. Basic Link Track Message Sets

Link tracks cannot be transmitted using any existing OTH-G message set due to the nature of their data structure. The CIX Message specification adds LCTC (basic information) and XPOS (positional information) message sets to those already provided by OTH-G for track encoding.

Appendix B describes the LCTC message set and Appendix C describes the XPOS message set.  Examples of Link track messages:

```
LCTC/JT4012345678/F14/L11//1232/AIR/00/FRD/7/132//1212/2323
XPOS/01211534.5Z2/AUG95/LL:304055.5N2-1304055.5E3/RADAR/150.7T
/123.5NM/76.8NM/120.6T/23.5KTS//130FT

LCTC/MAR013456789//L16//1423/SUB/00/SUS/4/203
XPOS/023456Z0/AUG95/LL:304055.5N2-1304055.5E3/////135.9T/15KTS
```

### 3. Extended Link Track Message Set

Link tracks which contain information beyond the basic data is contained in the optional LEXT message set, described in Appendix D.

### 4. Theater Ballistic Missile Track Message Set

Theater Ballistic Missile tracks also do not lend themselves to transmission using any existing OTH-G message set, because of their data structure.  Appendix E describes the BMSIL message set.  Example of a BMISL message set:

```
BMISL/JT4012345678//3/HOS////////4/01211534.5Z2/AUG95
/LL:304055.5N2-
1304055.55E8/150.7T/123.5NM/76.8NM/120.6T/123.5KTS/75
/ALT150FT
```

### 5. XML Version of a Typical CIX Formatted Message

The XML formats used for CIX closely parallel the CIX message sets.  Each message field value is wrapped in XML tags that carry the name or abbreviation of the field.  For example, in the LCTC message set, field 1 is the "Unique ID" so the XML tag corresponding to this field would be <UID>.  Each set of fields which comprise a message set are, in turn, wrapped in a set of tags that carry the name of the message set, e.g., <LCTC> and </LCTC>.  The complete XML message begins with the standard XML header and a <CIX> tag identifying the XML data as a CIX message. Appendix H contains a table which maps each CIX message field, grouped by message set, to its corresponding XML tag. Thus, for the following standard CIX message:

```
BT
MSGID/CTG 81.0/CIX/0001/MAY
LCTC/JT4012345678/F14/L11/COMAIRSOUTH STANDING
/1232/AIR/00/FRD/1235/7//1212/2323
XPOS/01211534.5Z2/AUG95/LL:304055.5N2-1304055.5E3
/RADAR/150.7T/123.5NM/76.8NM/120.6T/235.0KTS/ALT90
ENDAT
BT
NNNN
```

The corresponding CIX message using the XML format is provided.

```
BT
MSGID/CTG 81.0/CIX/0001/MAY
<?xml version="1.0"?>
<CIX>
<LCTC>
<UID>JT4012345678</UID>
<LTDNAME>F14</LTDNAME>
      <LTYPE>L11</LTYPE>
      <NAME>COMAIRSOUTH STANDING</NAME>
      <LTN>1232</LTN>
      <LREPTYPE>AIR</LREPTYPE>
      <EXSIM>00</EXSIM>
      <THREAT>FRD</THREAT>
      <RU>1235</RU>
      <TQ>7</TQ>
      <MODE1></MODE1>
      <MODE2>1212</MODE2>
      <MODE3>2323</MODE3>
</LCTC>
<XPOS>
      <DTG>01211534.5Z2</DTG>
      <MOYR>AUG95</MOYR>
      <POSIT>LL:304055.5N2-1304055.5E3</POSIT>
      <SENSOR>RADAR</SENSOR>
      <BSMAJ>150.7T</BSMAJ>
      <LSMAJ>123.5NM</LSMAJ>
      <LSMIN>76.8NM</LSMIN>
      <CSE>120.6T</CSE>
      <SPD>235.0KTS</SPD>
      <ALT>ALT90</ALT>
</XPOS>
</CIX>
ENDAT
BT
NNNN
```

## C. XML-MTF MESSAGE MAPPING RELATIONSHIPS FOR WXOBS

One of the messages selected is the USMTF message WXOBS (Weather Observation). This message structure will be used to highlight four of the mapping specifications between XML-MTF message components and the analogous MTF message components. To illustrate this, given below is a WXOBS message followed by its XML equivalent. The structure of the USMTF WXOBS message is provided in Appendix I. In a typical message there are several sets, some of which are mandatory and some optional [MTF00, MTFWS]. The example shown here has only mandatory sets.

```
EXER/EAGLE_HUNT//
MSGID/WXOBS/FTKNOX//
WEATHLOC/4633S13645W//
OBSTIME/201733Z//
WIND/217/-/51//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  /  18/     23/   278/  72/
/M10KFT /  53/     58/   222/  52//
```

```
<exercise_identification>
  <exercise_nickname>EAGLE_HUNT</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>WXOBS
  </message_text_format_identifier>
  <originator>FTKNOX</originator>
</message_identification>
<location_of_weather>
  <latitude_degrees>46</latitude_degrees>
  <latitude_minutes>33</latitude_minutes>
  <latitude_hemisphere>S</latitude_hemisphere>
  <longitude_degrees>136</longitude_degrees>
  <longitude_minutes>45</longitude_minutes>
  <longitude_hemisphere>W</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>20</day>
  <time_hour>17</time_hour>
  <time_min>33</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>
<wind>
  <wind_direction>217</wind_direction>
  <variable_wind_direction/>
  <wind_speed>51</wind_speed>
</wind>
<5uperair_data>
  <5uperair_data_group_of_fields>
    <temperature_celsius> 18</temperature_celsius>
    <dewpoint_temperature_celsius> 23
    </dewpoint_temperature_celsius>
    <wind_direction>278</wind_direction>
    <peak_gusts_knots>72</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius> 53</temperature_celsius>
    <dewpoint_temperature_celsius> 58
    </dewpoint_temperature_celsius>
    <wind_direction>222</wind_direction>
    <peak_gusts_knots>52</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
```

## 1. Element Name Mapping

XML-MTF element names are constructed from MTF components. The components are processed according to standards. This process of reconstruction is also called “filtering”. Component filtering does not guarantee unique XML-MTF names. This is an issue related to Namespace. The following table gives a sample of element reconstuction.

```
EXER/EAGLE_HUNT//
MSGID/WXOBS/FTKNOX//

<exercise_identification>
  <exercise_nickname>EAGLE_HUNT</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>WXOBS
  </message_text_format_identifier>
  <originator>FTKNOX</originator>
</message_identification>
```

| MTF Set | MTF Element Format Spec | XML-MTF Element |
|---|---|---|
| EXER | EXER Set | exercise_identification |
|  | EXERCISE NICKNAME | exercise_nickname |
| MSGID | MSGID Set | message_identification |
|  |  |  |

### 2. Composite Field Mapping

An MTF element containing composite data is broken down into its parent and child elements in an XML-MTF mapped element. This can be clarified with the OBSTIME set in the WXOBS MTF message. The field element of OBSTIME corresponds to the child elements of the XML version.

```
OBSTIME/201733Z//

<observation_day_time>
  <day>20</day>
  <time_hour>17</time_hour>
  <time_min>33</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>

The structure for the field element is:
<FUD-name>data</FUD-name>
```

### 3. Empty Field Mapping

A mandatory MTF Field containing no data is mapped to an empty XML-MTF field element. Conditional and operationally determined Fields that contain no data are omitted from the XML-MTF message. The empty field (variable wind direction) in the WIND MTF message set is mapped to an empty field in the XML version.

```
WIND/217/-/51//

<wind>
  <wind_direction>217</wind_direction>
  <variable_wind_direction/>
  <wind_speed>51</wind_speed>
</wind>

The structure for the field element is:
<FUD-name>elementals</FUD-name>
```

### 4. Columnar Set Mapping

In Columnar Set Mapping, the content of an XML-MTF columnar set is mapped from the rows of data contained in the mapped MTF Columnar set. Each data row is mapped to a “Group of Fields” element. The overall content of the set is a series of “Group of Fields” elements. In the following example, each row of data forms Group of Fields element in the XML version.

```
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  /  18/     23/   278/  72/
/M10KFT /  53/     58/   222/  52//

<5uperair_data>
  <5uperair_data_group_of_fields>
    <altitude> M5KFT</altitude>
    <temperature_celsius> 18</temperature_celsius>
    <dewpoint_temperature_celsius> 23
    </dewpoint_temperature_celsius>
    <wind_direction>278</wind_direction>
    <peak_gusts_knots>72</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <altitude> M10KFT</altitude>
    <temperature_celsius> 53</temperature_celsius>
    <dewpoint_temperature_celsius> 58
    </dewpoint_temperature_celsius>
    <wind_direction>222</wind_direction>
    <peak_gusts_knots>52</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
```

THIS PAGE INTENTIONALLY LEFT BLANK

# VII. SOFTWARE DEVELOPMENT

## A. INTRODUCTION

The primary effort of this thesis is to support Young’s work in resolving data representational differences between heterogeneous systems with a tool to generate sets of selected messages to be used in evaluating Young’s Object Oriented Method for Interoperability [YBGL02].

Towards this goal, four legacy databases (AFATDS, JCDB, GCCS-I3 ISDS, and GCCS-TDBM) were evaluated. The GCCS-TDBM database was found to be the most suitable due to reasons of access, availability and timeliness of data. Next, the messaging formats were analyzed. Finally, four message sets were selected to be representative of the interaction to be encountered in an actual environment.

This chapter details the development of the software tool to be used to generate the messages.

## B. SOFTWARE REQUIREMENTS

Requirements for the desired tool were as follows:

- The tool should generate messages in the native format and also the XML format.
- The selected messages shall be from the GCCS-TDBM database.
- The input data for the messages can be any legal value as permissible.
- The messages shall have a random nature.
- The preferred developmental language is Ada.
- The deliverable shall be an executable (.exe) file.
- The output from the tools shall be used as a data file for further analysis.

## C. ASSUMPTIONS MADE FOR SOFTWARE DEVELOPMENT

Based on the requirements specified, certain assumptions were made to help in the software development:

- Since the output from the tool is not interfaced with any other program in real-time mode, the need for timing constraints is eliminated. Hence, multitasking algorithms were not implemented.
- The program was designed to generate the designated message sets only. It is not a general purpose program.
- Each message set shall comprise a module in the program to facilitate addition or deletion of message sets.
- The USMTF message and the CIX message each have a separate executable and output files.
- The messages were to be generated in both the native format and XML format in separate output files.
- The various parametric values required to populate the messages were hard coded as enumerated types.
- The generated messages contain only the mandatory fields.

## D. FUNCTIONAL SPECIFICATION

A functional specification is a black-box model of the proposed software system capturing just the aspects of its behavior relevant to the users of the system [BL90]. These models are important because they can be understood and analyzed without reference to the parts of the system. The items listed in Section VII.C above under assumptions along with the image below serve as the Functional Specification of the proposed system.

As seen in the Functional Specification System Diagram (Figure 1), two parallel paths have been defined for message generation; one for CIX format and the other for MTF format. Data from the suite of databases is fed into a module to prepare the message for either of the two formats. Then, the next module generates the required message.

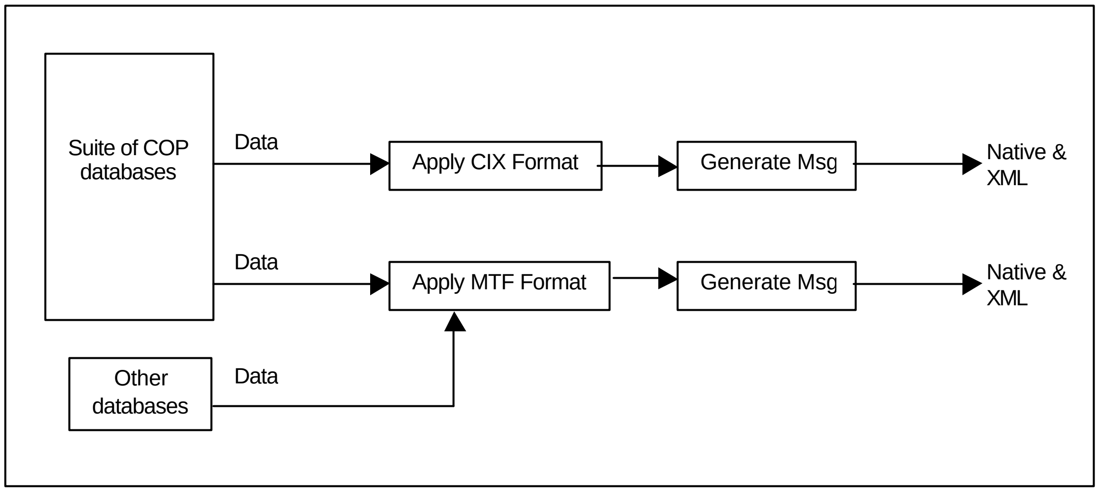

Figure 1: System Diagram

## E. ARCHITECTURAL DESIGN AND MODULE DEPENDENCY

An architectural design is a model of the proposed system that captures the aspects and behavior of the system relevant to the developer. It contains information needed by the developer to build the system. Hence, the goal here is to break up the proposed system into a set of small independent, self-contained units of code, called modules [BL90]. Modules can be decomposed at many levels of abstraction, until the specified submodules are simple enough to implement.

A module dependency diagram is useful for planning and future system evolution activities. It identifies the modules which must be implemented or simulated to test a given module, and which modules are affected by a change in a given module. The diagrams are useful for getting an overview of the system structure. Module dependency diagrams for both message structures are shown in Figures 2 and 3. The modules are described in paragraph VII.F.1 and VII.F.2.

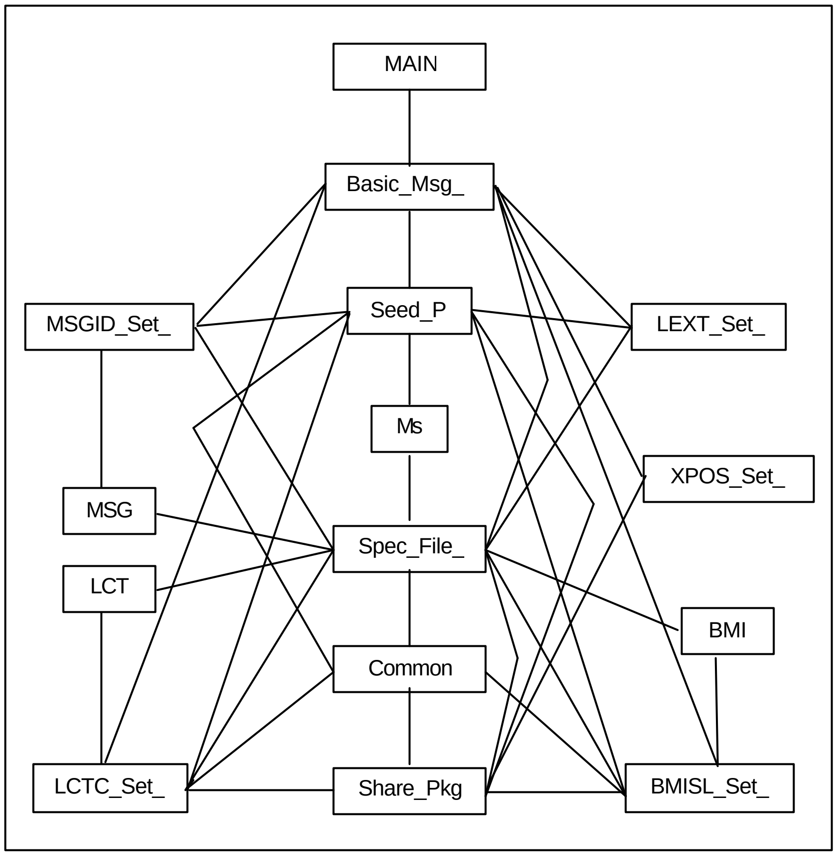

Figure 2: Module Dependency Diagram -CIX

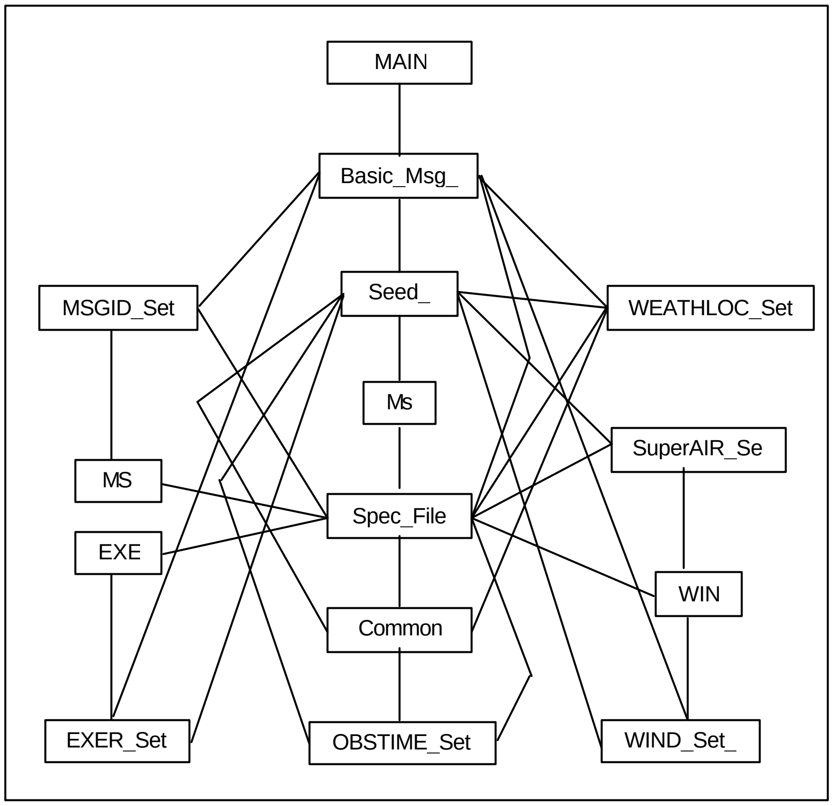

Figure 3: Module Dependency Diagram -MTF

## F. IMPLEMENTATION

An implementation is a description of the internal structure of each module specified in the architectural design. A completed implementation consists of an executable program for each module together with descriptions. One of the goals was to make certain that the program is as clear and simple as possible while conforming to the functional specification.

Modules can be used to model external systems such as users and peripheral devices, as well as software components as in this instance. The behavior of a model is specified by describing its interface. This means that all interactions must involve explicit instructions [BL90]. These principles were strictly adhered to in developing the modules for the program. A functional description of the modules follows. The source code for the CIX module is listed in Appendix J and the source code for the MTF module is listed in Appendix K.

**1. CIX Message Module Descriptions:**

- Main Module: This is the primary module. It sends commands to initialize the files, generate messages and finally close files.
- Basic_Msg_Pkg Module: This module contains the procedures to initialize and close files, add header and footer, and depending on the type of message to be generated, sends commands to other modules to build the appropriate portions of the message. The choice of messages are: Blink, Xlink and TBM. The message generator routine randomly chooses the type of message to be generated.
- MSGID_Set_Pkg Module: Message identification is required for messages of both formats. The content varies between them. This package generates the required message identification.
- LCTC_Set_Pkg Module: This generates Track information for the Basic Link Track Message set. The Link Type, Link Report Type, and Threat are randomly selected from an enumerated list. The Link Track Number is an alpha-numeric value randomly generated within the constraints specified.
- XPOS_Set_Pkg Module: This is also part of the Basic Link Track Message set. It generates positional information along with time-date information.
- LEXT_Set_Pkg Module: This is a part of the Extended Link Track Message set. There are no mandatory fields in this message set. Every field is

optional, and hence as per requirements, the fields are not simulated.

- BMISL_Set_Pkg Module: This is the Theater Ballistic Missile Set. It generates information on the track type, threat code, position and date-time information. The threat code is randomly selected from an enumerated list. The other data values are randomly generated within the constraints specified.
- Share_Pkg Module: Contains the common procedures used by the message set packages; namely to generate date-time, positional and threat information.
- Seed_Pkg Module: This module initializes all the parameters which require data to be randomized.
- Common/Msg/MSGID/LCTC/BMISL Modules: These modules are contained together in the Random_Files Module and is essential in generating the random values required for the various parameters.
- Spec_File_Pkg Module: All the type specifications with the ranges and enumerated lists are in this module.

**2. MTF Message Module Descriptions:**

- Main Module: This is the primary module. It sends commands to initialize the files, generate messages and finally close files.
- Basic_Msg_Pkg Module: This module contains the procedures to initialize and close files, add header and footer, and depending on the type of message to be generated, sends commands to other modules to build the appropriate portions of the message. The choice of messages in the MTF format are WXOBS, NBC1 and TARGET. The message generator routine randomly chooses the type of message to be generated. The

message generation routines for NBC1 and TARGET have not been implemented.

- MSGID_Set_Pkg Module: Message identification is required for messages of both formats. The content varies between them. This package generates the required message identification.
- EXER_Set_Pkg Module: This module is part of the WXOBS message. It generates the exercise identification information. The exercise code is randomly selected from an enumerated list.
- WEATHLOC_Set_Pkg Module: This module is part of the WXOBS message. It generates positional information for the particular point of interest.
- OBSTIME_Set_Pkg Module: This module is part of the WXOBS message. It generates the observed time information like day and time information.
- WIND_Set_Pkg Module: This module is part of the WXOBS message. It generates wind direction and speed information.
- SuperAIR_Set_Pkg Module: This module is part of the WXOBS message. This generates information on air conditions in the upper atmospheres. The data is formatted in “Columnar Set Mapping” wherein data is presented in rows of relevant information.
- Seed_Pkg Module: This module initializes all the parameters which require data to be randomized.
- Common/Msg/MSGID/EXER/WIND Modules: These modules are contained together in the Random_Files Module and is essential in generating the random values required for the various parameters.

- Spec_File_Pkg Module: All the type specifications with the ranges and enumerated lists are in this module.

## G. RUN TIME

After compilation and linking is completed, the executable files of interest are main.exe and main_mtf.exe. Running the executable produces two files; one for messages in the native format and the other for the same messages in XML format. Currently, the program is designed to generate 20 messages and can be easily changed to generate any required number. A sample of the run-time listing is in Appendix K.

## H. TESTING

Testing is an essential phase of software development. Testing helps in locating and diagnosing faults.

One category of errors that is easily eliminated during development include syntax errors, type consistency errors, and use of uninitialized variables. This was achieved by the use of the “Aonix Compiler” for Ada code development.

The amount of testing required for a module depends on the amount and complexity of its internal data. For modules with relatively small and simple data structures, like those in this program, extensive test instrumentation is not necessary. The code in this program is relatively simple with a few explicit loops and calls to other procedures.

As each of the message set package modules (MSGID, LCTC, XPOS, LEXT, BMISL, WXOBS) was completed, the module was individually compiled and executed to generate its message package.

Prior to coding of these message set package modules, they were rigged to send back hard coded messages without actually going through its actual algorithm. This enabled testing the skeletal structure of the system prior to a full scale development.

Much of the data in the system resides in the predefined components in the specification file. These predefined components were manually verified with the ranges specified for those variables. The parameters are listed in Appendix A – I.

The file operations were also verified by using the operating system for observing the file names created in the working directory.

## I. SOFTWARE REUSE

Software reuse is the integration and use of software assets from previously developed system rather than creating new versions for every similar application. When the USMTF message set was added to the requirement, it was not necessary to redesign the system; the existing CIX formatting packages were taken and modified easily to accommodate the USMTF.

## J. SOFTWARE EVOLUTION

This program encompasses a very small subset of the messages actually employed by the DOD. Eventually, it may become necessary to make modifications to this system to make it more valuable and useful to the user. The modules developed in this program can be easily reused with minor or no modifications.  This is illustrated in the next section.

## K. AN ILLUSTRATION OF SOFTWARE EVOLUTION AND REUSE

Consider the addition of a new CIX message set “CSITE”.  This message set contains basic site information common to both seaports and airfields. The CSITE message set will be accompanied by a CSEA message set, if the site is a seaport.  The CSITE and CSEA message sets are described in Appendices F and G, respectively.  Also, The CSITE/CSEA message sets will be accompanied by an XPOS message set, describing location data for the site.

Example of CSITE message:

```
CSITE/NORFOLK/US/23/4123
CSEA//SBEA/0/0/8/0/0
XPOS/01211534.5Z2/AUG95/LL:365100N5-0761800W2
```

The CSITE and CSEA message sets are currently not implemented in the message generation routine; the XPOS message set is already implemented and can be used.  To illustrate the implementation of the new message sets, the modifications to be made to the source document is shown in actual code or pseudo code form.

### 1. Changes to Basic_Msg_Pkg Module

The module Basic_Msg_Pkg.ada contains the list of messages to be generated. Statements are added to the case routine so that CSITE is one of the random messages chosen.

```
.
.
.
with CSITE_ set_pkg; -- added CSITE
with CSEA_set_pkg;   -- added CSEA
.
.
.
procedure Generate_Messages .....
.
.
.
      case Msg_set is
```

```
        .
        .
        .
        when spec_file_pkg.CSITE_TMS =>                     -- added CSITE message
          CSITE_set_Pkg.Add_CSITE_set (datafile, datafilex); -- to the case statement
          CSEA_set_Pkg.Add_CSEA_set (datafile, datafilex);
           XPOS_set_Pkg.Add_XPOS_set (datafile, datafilex);
         when others =>
        .
        .
        .
end Basic_Msg_Pkg;
```

## 2. Addition of New CSITE Module

The CSITE module (CSITE_set_pkg.ada) calls the appropriate routines to generate the CSITE message components as described by the rules and grammar listed in Appendix F. This module is very similar to the LCTC-set_pkg module and most of the code can be reused.

```
-- **************************
-- CSITE_set_pkg.ada
-- **************************
.
.
.
with CSITE;

package CSITE_set_Pkg is
  procedure Add_CSITE_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type);
  end CSITE_set_Pkg;

package body CSITE_set_Pkg is
  procedure Add_CSITE_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type) is
    Sitename     : spec_file_pkg.Sitename_type;
    Forcecode    : spec_file_pkg. Forcecode _type;
    UID          : spec_file_pkg.UID_type;
    --- and any other variables

begin

    --- add message leader in file “CSITE”

    --- add Sitename info to message
    Sitename := CSITE.sitename.Random(seed_pkg.G_site);
    CSITE.sitename_IO.put (datafile, Sitename);
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "      <NAME>");
    CSITE.sitename_IO.put (datafilex, Sitename);
    ada.Text_Io.Put (datafilex, "/NAME>");
    ada.Text_IO.new_line (datafilex);

    --- add Forcecode info to message
    .
    .
    .

    --- add UID info to message
    --- the UID generation can be reused from BMISL-set_pkg modules
    .
    .
    .
  end Add_CSITE_set;
end CSITE_set_Pkg;
```

### 3. Addition of new CSEA Module

This CSEA module (CSEA_set_pkg.ada) calls the appropriate routines to generate the CSEA message components as described by the rules and grammar listed in Appendix G. This module is very similar to the CSITE_set_pkg module described above and most of the code can be reused.

### 4. Reuse of XPOS Module

The third component in the CSITE message is the XPOS message set. This message set has been defined in the XPOS module (XPOS_set_pkg.ada). Hence, without an modification this module can be reused.

### 5. Additions to Random Module and Seed Module

The Random_files.ada contains the commands to enable the random choosing of variable values whenever appropriate. Similarly, the seed_pkg.ada file is also altered to include the seed for the appropriate variables.

```
-- **************************
-- Random_files.ada
-- **************************
.
.
.
package CSITE is
  package Sitename  is new Ada.Numerics.Discrete_Random(spec_file_pkg.Sitename_type);
  package Forcecode is new Ada.Numerics.Discrete_Random(spec_file_pkg.Forcecode_type);
.
.
.
end CSITE;

-- **************************
-- seed_pkg.ada
-- **************************
.
.
.
with CSITE;
with CSEA;

package body seed_pkg is
  procedure init_seed is
  begin
    .
    .
    .
    CSITE.sitename.Reset(G_name);
    CSITE.forcecode.Reset(G_code);
    .
    .
    .

  end init_seed;
end seed_pkg;
```

### 6. Additions to Spec Module

The Spec Module (spec_file_pkg.ada) contains all the variables with their type definitions, ranges and enumerations.

```
-- **************************
-- spec_file_pkg
-- **************************
package spec_file_pkg is

  type Msg_set_type is (BLink_TMS,
                        XLink_TMS,
                        TBM_TMS,
                        CSITE_TMS);

  type Sitename_type is  (LANGLEY, KNOX, .....);
  type Forcecode_type is Integer Range 1..39;
  .
  .
  .
  .

end spec_file_pkg;
```

## L. FUTURE MODIFICATIONS

It is very likely that this program will be modified later to suit the needs of the user. This modification may be the result of a change in the requirements, additional functionality, or errors in this implementation.  The most likely modification for this program will be addition of message sets, which was illustrated in the previous section.

# VIII. CONCLUSION

The primary effort of this thesis is to support Young’s work in resolving data representational differences between heterogeneous systems with a tool to generate sets of selected messages by analyzing several legacy databases and messaging formats.

The initial scope of work involved evaluating the following four legacy databases JCDB, AFATDS, GCCS I3 ISDS, and GCCS TDBM to determine a suitable subset of message definitions for random generation. On evaluating these, due to the difficulties encountered in obtaining real data from the four databases, (for reasons ranging from security issues to access and unavailability of data), the GCCS-TDBM was chosen as a representative database.

Several messaging formats employed for message transmission were analyzed. The USMTF and CIX formats were selected for message generation in their native and XML formats. Next, the messaging formats were analyzed. Finally, five message sets were selected to be representative of the interaction to be encountered in an actual environment.

Using software engineering principles, two programs were developed to generate USMTF and CIX formatted messages. This thesis contains the source code listing of these two programs in Appendices J and K. Also included is a sample listing of the messages generated by the two programs in both the native and XML formats in Appendices L and M. The output of this software program was used by Young to further his research work.

A description on how to add messages to the program, with an example, is also included.  As the needs of the user changes, the example provided would assist in making modifications and enhancements to the program.

# APPENDIX A: MSGID MESSAGE SET

Note: The "Use" column refers to whether the field is Mandatory or Optional. Also, the "Size/Type" column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 5AN | This field contains the literal string MSGID |
| 1 | Command | M | 1-14ANBS | Enter the command originating the message, e.g., CTF 70, COMTHIRDFLT, KITTY HAWK |
| 2 | Message ID | M | 3-9AN | Enter the message identifier: CIX |
| 3 | Message Serial Number | M | 4-5N | Enter the message serial number (MSN) (0001-9999), e.g., 0001, 0445, 9999. The MSN will always consist of 4 digits; the fifth is reserved for future use. |
| 4 | Month | M | 3A | Enter the first three letters of the month, e.g., JAN, FEB, MAR. |
| 5 | Operation/ Exercise Name | O | 1-20ANBS | Enter the operation or exercise name, or a brief description of the mission, as directed, e.g., PACEX 89. *Note: This field is not currently implemented.* |
| 6 | Qualifier | O | 3A | Enter the qualifier which further defines this message from the following list:<br>AMP Amplifies a previously sent message.<br>DEV Used to indicate a deviation from a previously sent message.<br>PER Signifies a message that is standing order for a lengthy period.<br>REQ Used to request a particular message, such as a tasking message, from another commander.<br>*Note: This field is not currently implemented..* |
| 7 | Qualifier Serial Number | O | 3N | Enter the serial number of the qualifier in Field 6. This number is used to uniquely identify qualifiers of a basic message, e.g., 001, 005, 999. *Note: This field is not currently implemented.* |

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX B: LCTC MESSAGE SET

Note: The "Use" column refers to whether the field is Mandatory or Optional. Also, the "Size/Type" column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 4AN | This field contains the literal string LCTC |
| 1 | Unique ID | O | 12AN | A unique identifier for the track, which will be present on tracks received from the GCCS-COP, but should be left NULL on transmission to the GCCS-COP. |
| 2 | Link Track Display Name | O | 1-10ANBS | Enter the name to be displayed for the link track, e.g., F14, Yorktown. |
| 3 | Link Type | M | 3-4AN | Enter the link type using the list below, e.g., L11, L16.<br><br>L11: Link-11 (TADIL A)<br>L11B: Link-11B (TADIL B)<br>L16: Link-16 (TADIL J)<br>L4A: Link-4A (TADIL C)<br>L22: Link-22 (NATO)<br><br>*Note: Currently only L11 and L16 are implemented* |
| 4 | Link Name | O | 1-20ANBS | Enter the name to identify the link, e.g. COMAIRSOUTH STANDING. *Note: This field is not currently implemented.* |
| 5 | Link Track Number | M | 4-5AN | Enter the link track number for the track being reported, e.g., 1234, 01012. |
| 6 | Link Report Type | M | 3-10ANBS | Enter the link track report type, e.g. AIR, SUB. |
| 7 | Exercise/ Simulation Indicators | M | 2N | Enter the exercise indicator (0 or 1) immediately followed by the simulation indicator (0 or 1) as provided in the link report, e.g., 10, 11, 00. The exercise indicator is set to 1 for an exercise track and is 0 otherwise. The simulation indicator is set to 1 for a simulated track and is zero otherwise. |
| 8 | Threat | M | 3A | Enter the treat code from the list below, e.g. FRD, AFD, SUS.<br><br><u>Code</u> <u>Description</u><br>PND Pending<br>UNK Unknown (Evaluated)<br>FRD Friend<br>AFD Assumed Friend<br>NEU Neutral<br>SUS Suspect<br>HOS Hostile |

| | | | | |
|---|---|---|---|---|
| 9 | RU (Reporting Unit) | O | 1-2N | Enter the RU (Reporting Unit), 4-5 octal digits, e.g. 1235, 4526, 11453. |
| 10 | Track Quality | O | 3-5N | Enter the track quality reported on the Link, (0-15), e.g. 5, 8. |
| 11 | Mode 1 IFF | O | 2N | Enter the Mode 1 IFF reported on the Link 01-73 octal (least significant digit cannot be greater than 3), e.g., 03 ,21, 42. |
| 12 | Mode 2 IFF | O | 4N | Enter the Mode 2 IFF reported on the Link, 4 octal digits (0001-7777), e.g., 1235, 5647, 1115. |
| 13 | Mode 3 IFF | O | 4N | Enter the Mode3 IFF reported on the Link, 4 octal digits (0001-7777), e.g., 1235, 5647, 1115. |
| 14 | Mode 4 IFF | O | 1N | Enter the Mode 4 IFF reported on the Link, (0-3), e.g., 1,3. |
| 15 | Discrete Identifier (DI) | O | 4N | Enter the DI for the track reported on the Link, 4 octal digits (0001-7777), e.g. 1234,5345. |
| 16 | CTSL | O | 3-5N | Enter the combat system local track number, e.g., 547, 1223. |
| 17 | Misc. Status | O | 3N | Enter, in sequence, the value of each of the following miscellaneous indicators reported on the link: Force Tell Indicator, Emergency Indicator, Special Processing Indicator, e.g., 100, 010, 011. Each indicator is reported as a value of 0 or 1. *Note: This field is not currently implemented.* |

# APPENDIX C: XPOS MESSAGE SET

Note: The “Use” column refers to whether the field is Mandatory or Optional. Also, the “Size/Type” column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 4AN | This field contains the literal string XPOS |
| 1 | Date-Time Group | M | 8-12ANS | Enter the date-time group of the position report in days (01-31), hours (00-23), minutes (00-59), and optional seconds (00-59) and tenths of seconds followed by the timezone (Z), and checksum (0-9). E.g., 01211534.5Z2, 212359Z3. |
| 2 | Month-Year | M | 5AN | Enter the first three letters of the month and last two digits of the year of the position report. e.g., JUN95, JAN01. |
| 3 | Position | M | 7-27ANS | Enter the position of the report in its original format and precision if possible. Use one of the alternate field contents provided below. Enter the designated field descriptor followed by the data. Data can be expressed in:<br><br><u>Coordinate System</u> — <u>Field Descriptor</u><br>Latitude/Longitude — LL:<br>UTM (Universal Transverse Mercator) — UT:<br>GEOREF (World Geographic Reference System) — GR:<br><br>The precision reported in this field should reflect the original known precision. The only boundary or restriction placed on the reported precision of data in this field is the field length range. An optional floating decimal is allowed as appropriate, e.g., LL:304055.55N7-1304055.55E8, UT:45FDK0474, GR:DIQA. *Note: Link tracks from the GCCS-COP will be transmitted in Lat/Long (LL:) format.* |
| 4 | Sensor Code | O | 2-6AN | Enter the sensor code of the detecting sensor, e.g., RADAR, VISUAL. See the *Operational Specification for Over-The-Horizon Targeting Gold, Rev. C, Ch. 1, Navy Center for Tactical Systems Interoperability, 01 August 1998,* Entry list 1104 (Sensor Codes) for more information. |
| 5 | Bearing of Major Axis | O | 4-6ANS | Enter the true bearing (000-360 or 000.0-360.0) of the semi-major axis followed by “T” (true), e.g., 5T, 135.5T. This field is mandatory if field 7 is used and is not equal to field 6. |
| 6 | Length of Semi-Major Axis | O | 2-7ANS | Enter the semi-major axis, in nautical miles (NM), kilometers (KM), meters (M), kiloyards (KY), yards (YD) or feet (FT). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure, e.g. 12.5NM, .045YD, 345KM. *Note: Link tracks* |

| | | | | |
|---|---|---|---|---|
| | | | | *from the GCCS-COP will be transmitted using Nautical Miles (NM).* |
| 7 | Length of Semi-Minor Axis | O | 2-7ANS | Enter the semi-minor axis, in nautical miles (NM), kilometers (KM), meters (M), kiloyards (KY), yards (YD) or feet (FT). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure, e.g. 12.5NM, .045YD, 345KM. *Note: Link tracks from the GCCS-COP will be transmitted using Nautical Miles (NM).* |
| 8 | Course | O | 4-6ANS | Enter the true course (000-360 or 000.0-360.0) of the track followed by "T" (true). e.g. 5T, 10.3T. |
| 9 | Speed | O | 4-8ANS | Enter the speed of the track in knots (KTS) or kilometers per hour (KPH). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure. e.g., 15KTS, 2.34KPH. *Note: Link tracks from the GCCS-COP will be transmitted using Knots (KTS).* |
| 10 | Altitude | O | 2-8ANS | Enter the altitude in hundreds of feet (ALT prefix), feet (FT suffix), kilometers (KM suffix), or meters (M suffix). Use up to 5 characters with optional floating decimal preceded or followed by appropriate prefix or suffix as provided in parenthesis following each unit of measure. e.g. ALT99 (for 9900 feet), 45.6M, 87FT. *Note: Link tracks from the GCCS-COP will be transmitted using feet (FT).* |
| 11 | Depth | O | 2-7ANS | Enter the depth in feet (FT), meters (M), kilometers (KM), or fathoms (FH). Use up to 5 characters with optional floating decimal followed by the appropriate suffix as provided in parenthesis following each unit of measure. e.g. .45FN, 12KM, 23FH. *Note: Link tracks from the GCCS-COP will be transmitted using feet (FT).* |
| 12 | RDF-RF | O | 3-10ANS | Enter the radio frequency of the intercept used to develop the XPOS in hertz (HZ), kilohertz (KHZ), or megahertz (MHZ). An optional floating decimal point is allowed, e.g., 5040.550HZ, 10050.5KHZ, 232.555MHZ. |
| 13 | Source Code | O | 2-6AN | Enter the source which most recently originated, passed or amplified data on the position being reported, e.g., OSIS, SELOR. See the *Operational Specification for Over-The-Horizon Targeting Gold, Rev. C, Ch. 1, Navy Center for Tactical Systems Interoperability, 01 August 1998*, Entry list 1104 (Source Codes) for more information. |

# APPENDIX D: LEXT MESSAGE SET

Note: The "Use" column refers to whether the field is Mandatory or Optional. Also, the "Size/Type" column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 4AN | This field contains the literal string LEXT |
| 1 | FOTC Track Number | O | 5N | Enter the track number used by correlation when operating in FOTC correlation mode. |
| 2 | CTSX Track Number | O | 4N | Enter the unique local link track number, assigned when a track enters the link. |
| 3 | System Track Number | O | 5N | Enter the System Track Number. This is also known as the Naval Tactical Display System (NTDS) number. |
| 4 | Tadil J Track Number | O | 5N | Enter the Tadil J Track Number. |
| 5 | Parent | O | 5N | Enter the local track number of a Platform track (if the Link track is associated with a Platform track). |
| 6 | AOU Type | O | 10ANS | Enter the Area of Uncertainty Type. Default type is an ELLIPSE with a semi-major and semi-minor axes of 9NM each. |
| 7 | AOU Bearing | O | 4ANS | Enter the AOU bearing for the track in degrees true. |
| 8 | XREF Code | O | 4AN | Enter the cross-reference code for the Command originating the track report. |
| 9 | Transmit Enabled | O | 1N | Enter the code indicating whether the track has been selected for transmission (enabled=1, disabled=0). |
| 10 | Received from Link | O | 1N | Enter the code indicating whether the track has been received from the Link (enabled--from the Link=1, disabled--not from the Link=0). |
| 11 | Emergency Status | O | 1N | Enter the code indicating whether the track has emergency status enabled (ON—overrides display filters; the track always displays and is transmitted at every opportunity=1. OFF—obeys display filters=0). |
| 12 | Force Tell Status | O | 1N | Enter the code indicating whether the track has Force Tell Status enabled (ON—overrides display filters=1. OFF—obeys display filters=0). |

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX E: BMISL MESSAGE SET

Note: The "Use" column refers to whether the field is Mandatory or Optional. Also, the "Size/Type" column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 5AN | This field contains the literal string BMISL |
| 1 | Unique ID | M | 12AN | A unique identifier for the track. *Note: This field is not currently implemented.* |
| 2 | Missile Type/ Generic Key | O | 4-20ANBS | Enter the missile type/ generic key. For missile reports received via TDDS in the TACELINT format, report the AEN in this field. For missile reports received via TDDS in the SENSOREP format, report the MIDB equipment code/ generic key in this field. For TIBS inputs report the character string which was decoded from the TDIMF mnemonic. For reports received over TADIL J, report the Space Specific Type. This information is not contained in reports received over CSI at this time. Field format is broad to accommodate TIBS literal. *Note: This field is not currently implemented.* |
| 3 | Track Type | M | 1N | Enter the track type code using the list below, e.g. 1, 2.<br><br><u>Code</u> <u>Description</u><br>1 Live Tactical<br>2 Live Training<br>3 Simulated Training |
| 4 | Threat | M | 3A | Enter the threat code from the list below, e.g. FRD, AFD, SUS.<br><br><u>Code</u> <u>Description</u><br>PND Pending<br>UNK Unknown (Evaluated)<br>FRD Friend<br>AFD Assumed Friend<br>NEU Neutral<br>SUS Suspect<br>HOS Hostile |
| 5 | Sensor | O | 2-6AN | Enter the sensor code of the detecting sensor from Entry list 1104 (Sensor Codes), e.g., ANSPY1, DSPIR (Enter DSPIR for IR sensors on Defense Support Program (DSP) satellites. Enter ANSPY1 for radar detections off the AEGIS AN/SPY1 radar.) *Note: This field is not currently implemented.* |
| 6 | Source | O | 2-6AN | Enter the source which most recently originated, passed or amplified data on the missile event being reported from Entry List 1136 (Source Codes), e.g. XO, CD. (For DSP detections enter the appropriate Correlation Index (CI). For |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  |  | AEGIS detections enter CD for AEGIS C&D/ACDS) |
| 7 | Source Track Number | O | 1-5AN | Enter the identifier or track number assigned by the source reported in field 6, e.g. 6032, 7132. For example, this could be the number assigned by the CI or the CTSX used by AEGIS. This field is optional and should be left blank if no identifier is assigned by the source. *Note: This field is not currently implemented.* |
| 8 | Network/ Format Received On | O | 4-6AN | Enter the code from the list below which represents the network and/or format in which the data was received.<br><br><u>Code</u> <u>Description</u><br>TRAPTE TRAP/TACELINT<br>TRAPS TRAP/SENSOREP<br>TIBS TIBS/TDIMF<br>CSIMT CSI/MT messages<br>TADILJ TADILJ<br>IBSNET INTEGRATED BROADCAST SERVICE NETWORK<br><br>*Note: This field is not currently implemented.* |
| 9 | Network ID/Track Number | O | 4-12AN | Enter the identifier or track number assigned by the network reported in field 8, e.g. 00123, 07001. For example enter the JTN for missile data received over TADIL J. If IBS decides to provide an identifier it could be reported in this field. This field is optional and should be left blank if no identifier is assigned by the network. *Note: This field is not currently implemented.* |
| 10 | MSC Status | O | 1N | Enter the Multi-Source Correlator (MSC) status from the list below, e.g. 1,3.<br><br><u>Code</u> <u>Description</u><br>0 Unprocessed<br>1 Processed<br>2 Correlated<br>3 Redundant<br>4 Fused<br><br>*Note: This field is not currently implemented. All BMISL messages transmitted will be for fused tracks.* |
| 11 | Time of Report or Event Seq. Number | O | 8-12ANS or 1-3N | Enter the date-time group of report if known; otherwise enter the internal system sequencing number associated with this event. When entering the date-time group of the report, enter days (01-31), hours (00-23), minutes (00-59), and optional seconds (00-59) and optional tenths of seconds followed by the timezone (Z), and checksum (0-9), e.g. 01120345Z6, 6. *Note: This field is not currently implemented.* |
| 12 | Type of Event | M | 1A | Enter the type of missile event from the list below, e.g. 1,3.<br><br><u>Code</u> <u>Description</u><br>1 Launch<br>2 Launch Update<br>3 Observation<br>4 Burn-out<br>5 Impact |

| | | | | |
|---|---|---|---|---|
| 13 | Event Date-Time Group | M | 8-12ANS | Enter the date-time group of the missile event in days (01-31), hours (00-23), minutes (00-59), and optional seconds (00-59) and optional tenths of seconds followed by the timezone (Z), and checksum (0-9). e.g., 01211534.5Z2, 212359Z3. |
| 14 | Event Month-Year | M | 5AN | Enter the first three letters of the month and last two digits of the year of the missile event. e.g., JUN95, JAN01. |
| 15 | Position | M | 7-27ANS | Enter the position of the missile event in its original format and precision if possible. Use one of the alternate field contents provided below. Enter the designated field descriptor followed by the data. Data can be expressed in:<br><br><u>Coordinate System</u> — <u>Field Descriptor</u><br>Latitude/Longitude — LL:<br>UTM (Universal Transverse Mercator) — UT:<br>GEOREF (World Geographic Reference System) — GR:<br><br>The precision reported in this field should reflect the original known precision. The only boundary or restriction placed on the reported precision of data in this field is the field length range. An optional floating decimal is allowed as appropriate, e.g., LL:304055.55N7-1304055.55E8, UT:45FDK0474, GR:DIQA. *Note: Link tracks from the GCCS-COP will be transmitted in Lat/Long (LL:) format.* |
| 16 | Bearing of Major Axis | O | 4-6ANS | Enter the true bearing (000-360 or 000.0-360.0) of the semi-major axis followed by "T" (true), e.g., 005T, 135.5T. This field is mandatory if field 18 is used and is not equal to field 17. |
| 17 | Length of Semi-Major Axis | O | 2-7ANS | Enter the semi-major axis, in nautical miles (NM), data miles (DM), kilometers (KM), meters (M), kiloyards (KY), yards (YD) or feet (FT). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure, e.g. 12.5NM, .045YD, 345KM. *Note: Link tracks from the GCCS-COP will be transmitted using Nautical Miles (NM).* |
| 18 | Length 0f Semi-Minor Axis | O | 2-7ANS | Enter the semi-minor axis, in nautical miles (NM), data miles (DM), kilometers (KM), meters (M), kiloyards (KY), yards (YD) or feet (FT). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure, e.g. 12.5NM, .045YD, 345KM. *Note: Link tracks from the GCCS-COP will be transmitted using Nautical Miles (NM).* |
| 19 | Course | O | 4-6ANS | Enter the true course (000-360 or 000.0-360.0) of the missile followed by "T" (true). e.g. 005T, 010.3T. |
| 20 | Speed | O | 4-8ANS | Enter the speed of the missile in knots (KTS), kilometers per hour (KPH), or kilometers per second (KPS). Use up to 5 characters with optional floating decimal (.0001-99999) followed by the appropriate alpha suffix as provided in parenthesis following each unit of measure. e.g., 15KTS, 2.34KPH. *Note: Link tracks from the GCCS-COP will be* |

| | | | | |
|---|---|---|---|---|
| | | | | *transmitted using knots (KTS).* |
| 21 | Angle of Elevation | O | 1-4NS | Enter the angle of elevation (0-90 or 0.0-90.0) of the missile. e.g. 5, 010.3. |
| 22 | Altitude | O | 2-8NS | Enter the altitude in hundreds of feet (ALT prefix), feet (FT suffix), kilometers (KM suffix), or meters (M suffix). Use up to 5 characters with optional floating decimal preceded or followed by appropriate prefix or suffix as provided in parenthesis following each unit of measure. e.g. ALT99 (for 9900 feet), 45.6M, 87FT. *Note: Link tracks from the GCCS-COP will be transmitted using feet (FT).* |

## APPENDIX F: CSITE MESSAGE SET

Note: The “Use” column refers to whether the field is Mandatory or Optional. Also, the “Size/Type” column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 4AN | This field contains the literal string CSITE |
| 1 | Site Name | M | 1-30ANBS | Enter the name of the site (e.g., LANGLEY AFB). If unknown, then enter as “UNKNOWN.” |
| 2 | Flag | O | 2A | Enter the flag of the site being reported from the OS-OTG Country Codes Entry List (e.g., US, UR). |
| 3 | Force Code | M | 2N | Enter the force code (01-39) of the site being reported as specified in the OS-OTG (REV-C) Change 1. |
| 4 | UID | M | 5-15AN | Enter a unique identifier (UID). Enter the identifier uniquely assigned by the system that introduces the site to the WAN. |

THIS PAGE INTENTIONALLY LEFT BLANK

## APPENDIX G: CSEA MESSAGE SET

Note: The “Use” column refers to whether the field is Mandatory or Optional. Also, the “Size/Type” column defines the data type and size. The allowable types are:

- Alphabetic (A): Upper case A through Z
- Numeric (N): 0 through 9
- Space (B): Space (ASCII 040 octal code)
- Special (S): Period (.), comma (,), colon (:), percent (%), pound (#), asterisk (*), hyphen (-), carriage return (CR), and line feed (LF).

For example, 2-6AN means the field can be from 2 to 6 alphanumeric characters, while 4N means the field must be a four digit number.

| Field | Field Name | Use | Size/ Type | Description |
|---|---|---|---|---|
| 0 | Set Type | M | 4AN | This field contains the literal string CSEA |
| 1 | MILSEA Code | O | 4A | Enter the four-character alphanumeric code representing the military-assigned MILSTAMP identifier or other host nation identifier for a seaport. |
| 2 | GEOLOC Code | O | 4A | Enter the four-character alphanumeric Geographic Location Code (e.g., FJXT, PYCT) as assigned by the Joint Chiefs of Staff (JCS). |
| 3 | Size | O | 1N | Enter the code (as recorded in Table 7) indicating whether the harbor is large, medium, small, very small, or unknown. |
| 4 | Max Vessel | O | 1N | Enter the code (as recorded in Table 8) indicating the maximum vessel size for the seaport. |
| 5 | Channel Depth | O | 1-2N | Enter the code (as recorded in Table 9) indicating the channel depth range in feet of the seaport. |
| 6 | Pier Depth | O | 1-2N | Enter the code (as recorded in Table 10) indicating the pier depth in feet at the pier or piers. |
| 7 | Tide | O | 1-5N | Enter the mean tide range in feet. |

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX H: CIX XML MESSAGING

The following table maps each CIX message field, grouped by message set, to its corresponding XML tag.

| Field | Field Name | XML Tag |
|---|---|---|
| **LCTC MESSAGE SET** | | |
| 0 | Set Type | <LCTC> |
| 1 | Unique ID | <UID> |
| 2 | Link Track Display Name | <LTDNAME> |
| 3 | Link Type | <LTYPE> |
| 4 | Link Name | <NAME> |
| 5 | Link Track Number | <LTN> |
| 6 | Link Report Type | <LREPTYPE> |
| 7 | Exercise/ Simulation Indicators | <EXSIM> |
| 8 | Threat | <THREAT> |
| 9 | RU (Reporting Unit) | <RU> |
| 10 | Track Quality | <TQ> |
| 11 | Mode 1 IFF | <MODE1> |
| 12 | Mode 2 IFF | <MODE2> |
| 13 | Mode 3 IFF | <MODE3> |
| 14 | Mode 4 IFF | <MODE4> |
| 15 | Discrete Identifier (DI) | <DI> |
| 16 | CTSL | <CTSL> |
| 17 | Misc. Status | <STATUS> |
| **XPOS MESSAGE SET** | | |
| 0 | Set Type | <XPOS> |
| 1 | Date-Time Group | <DTG> |
| 2 | Month-Year | <MOYR> |
| 3 | Position | <POSIT> |
| 4 | Sensor Code | <SENSOR> |
| 5 | Bearing of Major Axis | <BSMAJ> |
| 6 | Length of Semi-Major Axis | <LSMAJ> |
| 7 | Length of Semi-Minor Axis | <LSMIN> |
| 8 | Course | <CSE> |
| 9 | Speed | <SPD> |
| 10 | Altitude | <ALT> |
| 11 | Depth | <DPTH> |
| 12 | RDF-RF | <RDFRF> |
| 13 | Source Code | <SOURCE> |
| | | |
| | | |
| | | |

| Field | Field Name | XML Tag |
|---|---|---|
| **BMISL MESSAGE SET** | | |
| 0 | Set Type | <BMISL> |
| 1 | Unique ID | <UID> |
| 2 | Missile Type/ Generic Key | <MSLTP> |
| 3 | Track Type | <TTYP> |
| 4 | Threat | <THREAT> |
| 5 | Sensor | <SENSOR> |
| 6 | Source | <SOURCE> |
| 7 | Source Track Number | <SRCTN> |
| 8 | Network/ Format Received On | <NETFMT> |
| 9 | Network ID/Track Number | <NETID> |
| 10 | MSC Status | <MSC> |
| 11 | Time of Report or Event Seq. Number | <REPTME> |
| 12 | Type of Event | <REPTYP> |
| 13 | Event Date-Time Group | <EDTG> |
| 14 | Event Month-Year | <EMOYR> |
| 15 | Position | <POSIT> |
| 16 | Bearing of Major Axis | <BSMAJ> |
| 17 | Length of Semi-Major Axis | <LSMAJ> |
| 18 | Length 0f Semi-Minor Axis | <LSMIN> |
| 19 | Course | <CSE> |
| 20 | Speed | <SPD> |
| 21 | Angle of Elevation | <AOE> |
| 22 | Altitude | <ALT> |
| **LEXT MESSAGE SET** | | |
| 0 | Set Type | <LEXT> |
| 1 | Range | <RNG> |
| 2 | FOTC Track Number | <FTN> |
| 3 | CTSX Track Number | <CTSXTN> |
| 4 | System Track Number | <STN> |
| 5 | Tadil J Track Number | <TJTN> |
| 6 | Parent | <PRNT> |
| 7 | AOU Type | <AOUTYP> |
| 8 | AOU Bearing | <AOUBRG> |
| 9 | XREF Code | <XREF> |
| 10 | Transmit Enabled | <XMITENBL> |
| 11 | Received from Link | <RFL> |
| 12 | Emergency Status | <EMERSTAT> |
| 13 | Force Tell Status | <FTS> |

# APPENDIX I: WXOBS STRUCTURE

The structure of the USMTF WXOBS message as defined by the Joint User Handbook is provided here.

```
SEG    OCC RPT SETID

        C
EXER/________________________/...........................................//
                   1 EXERCISE NICKNAME      2 EXERCISE MESSAGE ADDITIONAL IDENTIFIER
                   M  [1-56 ANBS]           O  [1-16 ANBS]

                   Set 1 Purpose: THE EXER SET PROVIDES THE DESIGNATED
                   CODE NAME OR NICKNAME, IF THE MESSAGE

                   SUPPORTS AN EXERCISE.

                   Note: THE EXER SET IS PROHIBITED IF THE OPER SET IS USED.

        O
OPER/_________________________/................................./....................../.
...............................//
                   1 OPERATION CODEWORD      2 PLAN ORIGINATOR AND NUMBER      3 OPTION
NICKNAME      4 SECONDARY OPTION NICKNAME

                   M  [1-32 ANBS]            O  [5-36 ANBS]                    O  [1-23
ANBS]         O  [1-23 ANBS]

                   Set 2 Purpose: THE OPER SET PROVIDES THE DESIGNATED
                   CODE NAME OR NICKNAME, IF THE MESSAGE
                   SUPPORTS AN OPERATION.

                   Note: THE OPER SET IS PROHIBITED IF THE EXER SET IS USED.

        M
MSGID/_____________________________________/__________________/...........................
./................/................/................................//

                    1 MESSAGE TEXT FORMAT IDENTIFIER      2 ORIGINATOR      3 MESSAGE
SERIAL NUMBER      4 MONTH NAME      5 QUALIFIER      6 SERIAL NUMBER OF QUALIFIER
                    M  [2-20 ANBS]                        M  [1-30 ANBS]    O  [1-7 ANBS]
O  [3 A]          O  [3 A]         O  [1-3 N]

                    Set 3 Purpose: THE MSGID SET PROVIDES THE MESSAGE

                    IDENTIFICATION AND ORIGINATOR.

                    Note: FIELD 1 OF THE MSGID SET MUST EQUAL "WXOBS".

        O   R
REF/____________________/________________________/_________________/_____________________
________________/................................./......................../..............
R................//
                  1 SERIAL LETTER      2 TYPE OF REFERENCE      3 ORIGINATOR      4 DATE
AND/OR TIME OF REFERENCE      5 SERIAL NUMBER OF REFERENCE      6 SPECIAL NOTATION      7
SIC CODE OR FILING NUMBER

                  M  [1 A]             M  [2/20 ANBS]           M  [1-30 ANBS]    M
[6/15 AN]                    O  [1-10 ANBS]                    O  [5 A]
O  [1/10 ANBS]

                  Set 4 Purpose: THE REF SET PROVIDES BOTH USMTF AND
                  NON-USMTF REFERENCES.


        C      AMPN/________________//
                   1 FREE TEXT
                   M  [1-U XEL]

                   Set 5 Purpose: THE AMPN SET PROVIDES ADDITIONAL
                   INFORMATION ON THE PRECEDING REF SET.
                   ADDITIONALLY, THE AMPN SET PROVIDES
                   IDENTIFYING INFORMATION FOR A NON-USMTF
                   REFERENCE.

Note: THE AMPN SET IS MANDATORY IF FIELD 2 OF THE REF SET IS A OMMUNICATION  TYPE AND
ONLY ONE REFERENCE IS USED.

        C      NARR/________________//
                   1 FREE TEXT
                   M  [1-U XEL]

                   Set 6 Purpose: THE NARR SET PROVIDES ADDITIONAL
                   INFORMATION ON THE PRECEDING TWO OR MORE
                   REF SETS.  ADDITIONALLY, THE NARR SET
                   PROVIDES IDENTIFYING INFORMATION FOR
                   NON-USMTF REFERENCES.

Note: THE NARR SET IS MANDATORY IF THE REF SET IS REPEATED ONE OR MORE TIMES AND FIELD 2
OF ONE OR MORE REF SETS CITES A COMMUNICATION TYPE.

M       M      WEATHLOC/__________________________//
|                       1 LOCATION OF WEATHER
|                       M  [1/38 ANBS]
|
|                       Set 7 Purpose: THE WEATHLOC SET PROVIDES THE LOCATION

|                       OF THE OBSERVED WEATHER.
|
|                       OBSERVATIONS FOR MULTIPLE AREAS.
|
|       M      OBSTIME/___________________________//
|                      1 OBSERVATION DAY-TIME
|                      M  [7 AN]
|
|                      Set 8 Purpose: THE OBSTIME SET PROVIDES THE TIME OF THE

|                      WEATHER OBSERVATION.
|
|
|       O
CLDLYR/______________________/______________R______________/________R________/___________
______R_________________//
|                     1 MINIMUM CEILING      2 CLOUD COVER IN EIGHTHS      3 CLOUD TYPE
4 ALTITUDE IN HUNDREDS OF FEET
|                     M  [1-3 N]             M  [1 N]                      M  [2 A]
M  [1-3 N]

|
|                     Set 9 Purpose: THE CLDLYR SET PROVIDES THE OBSERVED
|                     CLOUD LAYER INFORMATION.
|
|
|       O      VISBY/_________________/...........R............//
|                    1 VISIBILITY      2 WEATHER, STATE OF
|                    M  [2-7 ANS]      O  [3-10 ABS]
|
|                    Set 10 Purpose: THE VISBY SET PROVIDES THE OBSERVED

|                    VISIBILITY.
|
|
|       O
TEMP/___________________________________/______________________________________________/.
..................................................//
|                  1 MAXIMUM TEMPERATURE, CELSIUS      2 MINIMUM TEMPERATURE, IN DEGREES
CELSIUS      3 MINIMUM FREEZING LEVEL IN THOUSANDS OF FEET
```

```
|                  M  [1-4 NS]                          M  [1-4 NS]
O  [4-7 AN]

|
|                  Set 11 Purpose: THE TEMP SET PROVIDES THE OBSERVED
|                  TEMPERATURE.
|
|
|       O
WIND/_______________________________________________________/...........................
............./_____________________________/.........................//
|                  1 WIND DIRECTION IN DEGREES RELATIVE TO TRUE NORTH      2 VARIABLE
WIND DIRECTION IN DEGREES      3 SPEED OF WIND IN KNOTS      4 PEAK GUSTS, IN KNOTS

|                  M  [3 N]                                                O  [3 N]
M  [1-3 N]                O  [1-3 N]
|
|                  Set 12 Purpose: THE WIND SET PROVIDES THE OBSERVED WIND
|                  INFORMATION.
|
|
|       O      ALTSTG/___________________________________________________________//
|                    1 ALTIMETER SETTING IN HUNDREDTHS OF INCHES OF MERCURY

|                    M  [4 N]
|
|                    Set 13 Purpose: THE ALTSTG SET PROVIDES THE OBSERVED
|                    ATMOSPHERIC PRESSURE INFORMATION.
|
|
|       O      5UPERAIR
|              /ALT            /TMPC                            /DEWTEMP
/WINDIR                                              /SPD
//

|
/_______________/____________________________________/__________________________________
___________/......................................................./....................
......................//
|              1 ALTITUDE      2 REPORTED TEMPERATURE, CELSIUS      3 DEWPOINT
TEMPERATURE IN DEGREES CELSIUS      4 WIND DIRECTION IN DEGREES RELATIVE TO TRUE NORTH
5 SPEED IN KNOTS, DECIMAL PT PERMITTED
|              M  [3-7 AN]     M  [1-4 NS]                      M  [1-4 NS]
O  [3 N]                                             O  [1-4 NS]

|
|              Set 14 Purpose: THE 5UPERAIR SET PROVIDES THE OBSERVED
|              UPPER LEVEL WINDS.
|
|
        O
DECL/________________________________/............................../................R.
................/........................R........................//
                   1 SOURCE FOR CLASSIFICATION      2 REASON FOR CLASSIFICATION      3
DOWNGRADE INSTRUCTIONS/DATE      4 DOWNGRADING OR DECLASSIFICATION EXEMPTION CODE

                   M  [1/55 ANBS]                 C  [3 AN]                        C
[1/38 ANBS]                 C  [2 AN]

                   Set 15 Purpose: THE DECL SET PROVIDES DECLASSIFICATION
                   OR DOWNGRADING INSTRUCTIONS, IF THE
                   MESSAGE IS CLASSIFIED.
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX J: SOURCE LISTING FOR CIX FORMATTED MESSAGE

```
-- *******************
-- Main.ada
-- *******************

with ada.text_io;
with Basic_Msg_Pkg;

procedure main is
  datafile : ada.text_io.file_type;
  datafilex: ada.text_io.file_type;

begin
  Basic_Msg_Pkg.Initialize (datafile, datafilex);
  Basic_Msg_Pkg.Generate_Messages (datafile,datafilex);
  Basic_Msg_Pkg.Closefile (datafile,datafilex);
end main;

-- ******************************
-- Basic_Msg_Pkg.ada
-- ******************************

with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with Msg;
with MSGID_set_pkg;
with LCTC_set_pkg;
with XPOS_set_pkg;
with LEXT_set_pkg;
with BMISL_set_pkg;

package Basic_Msg_Pkg is
  procedure Initialize (datafile:  out ada.text_io.file_type;
                        datafilex: out ada.text_io.file_type);
  procedure Add_Header (datafile:  in out ada.text_io.file_type;
                       datafilex:  in out ada.text_io.file_type);
  procedure Add_Footer (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type);
  procedure Closefile  (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type);
  procedure Generate_TestMessages (datafile:  in out ada.text_io.file_type;
                                   datafilex: in out ada.text_io.file_type);
  procedure Generate_Messages (datafile:  in out ada.text_io.file_type;
                               datafilex: in out ada.text_io.file_type);
end Basic_Msg_Pkg;

package body Basic_Msg_Pkg is
  procedure Initialize (datafile:  out ada.text_io.file_type;
                        datafilex: out ada.text_io.file_type) is
  begin
    seed_pkg.init_seed;
    ada.Text_Io.Create (File=> datafile,
                        Mode=> Ada.Text_Io.out_file,
                        Name=> "A:\Msg_File_n.txt");
    ada.Text_Io.Create (File=> datafilex,
                        Mode=> Ada.Text_Io.out_file,
                        Name=> "A:\Msg_File_x.txt");
  end Initialize;

  procedure Add_Header (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put_Line (datafile,  "BT");
    ada.Text_Io.Put_Line (datafilex, "BT");
  end Add_Header;
  procedure Add_Footer (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put_Line (datafile,  "ENDAT");
    ada.Text_Io.Put_Line (datafile,  "BT");
    ada.Text_Io.Put_Line (datafilex, "</CIX>");
    ada.Text_Io.Put_Line (datafilex, "ENDAT");
    ada.Text_Io.Put_Line (datafilex, "BT");
    for i in 1..7 loop
      ada.text_io.New_Line (datafile);
      ada.text_io.New_Line (datafilex);
    end loop;
    ada.text_io.Put_Line (datafile,  "NNNN");
    ada.text_io.Put_Line (datafilex, "NNNN");
  end Add_Footer;

  procedure Closefile (datafile:  in out ada.text_io.file_type;
                       datafilex: in out ada.text_io.file_type) is
  begin
    ada.text_io.close (datafile);
    ada.text_io.close (datafilex);
  end Closefile;

  procedure Generate_TestMessages (datafile:  in out ada.text_io.file_type;
                                   datafilex: in out ada.text_io.file_type) is
  begin
    for I in 1 .. 20 loop
      Add_Header (datafile, datafilex);
      MSGID_set_Pkg.Add_MSGID_set (datafile, datafilex);
      LCTC_set_Pkg.Add_LCTC_set (datafile, datafilex);
      XPOS_set_Pkg.Add_XPOS_set (datafile, datafilex);
      LEXT_set_Pkg.Add_LEXT_set (datafile, datafilex);
      Add_Footer (datafile, datafilex);
     end loop;
  end Generate_TestMessages;

  procedure Generate_Messages (datafile:  in out ada.text_io.file_type;
                               datafilex: in out ada.text_io.file_type) is
  Msg_set : spec_file_pkg.Msg_set_type;
  begin
    for I in 1..20 loop
      Msg_set := Msg.Random(seed_pkg.G_msg);
      Add_Header (datafile, datafilex);
      MSGID_set_Pkg.Add_MSGID_set (datafile, datafilex);
      case Msg_set is
        when spec_file_pkg.BLink_TMS =>
          LCTC_set_Pkg.Add_LCTC_set (datafile, datafilex);
          XPOS_set_Pkg.Add_XPOS_set (datafile, datafilex);
        when spec_file_pkg.XLink_TMS =>
          LCTC_set_Pkg.Add_LCTC_set (datafile, datafilex);
          XPOS_set_Pkg.Add_XPOS_set (datafile, datafilex);
          LEXT_set_Pkg.Add_LEXT_set (datafile, datafilex);
        when spec_file_pkg.TBM_TMS =>
          LCTC_set_Pkg.Add_LCTC_set (datafile, datafilex);
          BMISL_set_Pkg.Add_BMISL_set (datafile, datafilex);
        when others =>
          null;
      end case;
      Add_Footer (datafile, datafilex);
    end loop;
  end Generate_Messages;

end Basic_Msg_Pkg;


-- **************************
-- MSGID_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with MSGID;

package MSGID_set_Pkg is
  procedure Add_MSGID_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type);
end MSGID_set_Pkg;

package body MSGID_set_Pkg is
  procedure Add_MSGID_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type) is
    Command   : spec_file_pkg.Command_type;
    Serialnum : spec_file_pkg.Serialnum_type;
    Month     : spec_file_pkg.Month_type;

  begin
    ada.Text_Io.Put (datafile,  "MSGID/");
    ada.Text_Io.Put (datafilex, "MSGID/");

    Command := MSGID.Command.Random(seed_pkg.G_com);
    MSGID.Command_IO.put (datafile,  Command);
    MSGID.Command_IO.put (datafilex, Command);

    ada.Text_Io.put (datafile,  "/CIX/");
    ada.Text_Io.put (datafilex, "/CIX/");

    Serialnum := MSGID.Serialnum.Random(seed_pkg.G_ser);
    MSGID.Serialnum_IO.put (datafile,  Serialnum, 4);
    MSGID.Serialnum_IO.put (datafilex, Serialnum, 4);

    ada.Text_Io.put (datafile,  "/");
    ada.Text_Io.put (datafilex, "/");

    Month := MSGID.Month.Random(seed_pkg.G_mon);
    MSGID.Month_IO.put (datafile,  Month);
    MSGID.Month_IO.put (datafilex, Month);
    ada.text_io.new_line (datafile);
    ada.text_io.new_line (datafilex);
    ada.Text_Io.put_line (datafilex, "<CIX>");
  end Add_MSGID_set;

end MSGID_set_Pkg;



-- **************************
-- LCTC_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with LCTC;
with common;
with share_pkg;

package LCTC_set_Pkg is
  procedure Add_LCTC_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type);
  end LCTC_set_Pkg;

package body LCTC_set_Pkg is
  procedure Add_LCTC_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type) is
    Link        : spec_file_pkg.Link_type;
    Tracknum    : spec_file_pkg.Tracknum_type;
    TracknumA   : spec_file_pkg.TracknumA_type;
    Link_Report : spec_file_pkg.Link_Report_type;
    Exer_Sim_Ind : spec_file_pkg.Exer_Sim_Ind_type;
    temp        : String(1..3);
  begin
```

```
    ada.Text_Io.Put (datafile,  "LCTC///");
    ada.Text_Io.Put (datafilex, "<LCTC>");
    ada.Text_IO.new_line (datafilex);

    Link := LCTC.link.Random(seed_pkg.G_lin);
    LCTC.link_IO.put (datafile, Link);
    ada.Text_Io.put (datafile, "//");
    ada.Text_Io.Put (datafilex, "      <LTYPE>");
    LCTC.link_IO.put (datafilex, Link);
    ada.Text_Io.Put (datafilex, "/LTYPE>");
    ada.Text_IO.new_line (datafilex);

    Tracknum := LCTC.tracknum.Random(seed_pkg.G_tra);
    LCTC.tracknum_IO.put (datafile, Tracknum, 4);
    TracknumA := LCTC.tracknumA.Random(seed_pkg.G_trB);
    temp := spec_file_pkg.TracknumA_type'Image(TrackNumA);
    ada.text_io.put(datafile,temp(2));
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "      <LTN>");
    LCTC.tracknum_IO.put (datafilex, Tracknum, 4);
    ada.text_io.put (datafileX,temp(2));
    ada.Text_Io.Put (datafilex, "/LTN>");
    ada.Text_IO.new_line (datafilex);

    Link_Report := LCTC.report.Random(seed_pkg.G_rep);
    LCTC.report_IO.put (datafile, Link_Report);
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "      <LREPTYPE>");
    LCTC.report_IO.put (datafilex, Link_Report);
    ada.Text_Io.Put (datafilex, "/LREPTYPE>");
    ada.Text_IO.new_line (datafilex);

    ada.Text_Io.Put (datafilex, "      <EXSIM>");
    for I in 1 .. 2 loop
       Exer_Sim_Ind := LCTC.exersim.Random(seed_pkg.G_exe);
       LCTC.exersim_IO.put (datafile,  Exer_Sim_Ind, 1);
       LCTC.exersim_IO.put (datafilex, Exer_Sim_Ind, 1);
    end loop;
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "/EXSIM>");
    ada.Text_IO.new_line (datafilex);

    Share_pkg.threatcode (datafile, datafilex);

    ada.Text_IO.new_line (datafile);
    ada.Text_Io.Put (datafilex, "</LCTC>");
    ada.Text_IO.new_line (datafilex);

  end Add_LCTC_set;
end LCTC_set_Pkg;

-- **************************
-- XPOS_set_pkg.ada
-- **************************
with ada.text_io;
with share_pkg;

package XPOS_set_Pkg is
  procedure Add_XPOS_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type);
end XPOS_set_Pkg;

package body XPOS_set_Pkg is
  procedure Add_XPOS_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put (datafile,  "XPOS/");
    ada.Text_Io.Put (datafilex, "<XPOS>");
    ada.Text_IO.new_line (datafilex);

    Share_pkg.DateTimeGroup (datafile, datafilex);
    ada.Text_Io.Put (datafile, "/");

    Share_pkg.MonthYearField (datafile, datafilex);
    ada.Text_Io.Put (datafile, "/");

    Share_pkg.PositionField (datafile, datafilex);
    ada.Text_IO.new_line (datafile);

    ada.Text_Io.Put (datafilex, "</XPOS>");
    ada.Text_IO.new_line (datafilex);

  end Add_XPOS_set;
end XPOS_set_Pkg;

-- **************************
-- LEXT_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;

package LEXT_set_Pkg is
  procedure Add_LEXT_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type);
end LEXT_set_Pkg;

package body LEXT_set_Pkg is
  procedure Add_LEXT_set (datafile:  in out ada.text_io.file_type;
                          datafileX: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put (datafile, "LEXT");
    ada.Text_IO.new_line (datafile);
    ada.Text_Io.Put (datafilex, "<LEXT>");
    ada.Text_IO.new_line (datafilex);
    ada.Text_Io.Put (datafilex, "</LEXT>");
    ada.Text_IO.new_line (datafilex);
  end Add_LEXT_set;
end LEXT_set_Pkg;

-- **************************
-- BMISL_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with BMISL;
with common;
with share_pkg;

package BMISL_set_Pkg is
  procedure Add_BMISL_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type);
end BMISL_set_Pkg;

package body BMISL_set_Pkg is
  procedure Add_BMISL_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type) is
    UID1        : spec_file_pkg.UID1_type;
    UID2        : spec_file_pkg.UID2_type;
    Trk         : spec_file_pkg.Trk_type;
    Threat_Code : spec_file_pkg.Threat_Code_type;
    Event       : spec_file_pkg.Event_type;
    temp        : string(1..3);

  begin
    ada.Text_Io.Put (datafile,  "BMISL/");
    ada.Text_Io.Put (datafilex, "<BMISL>");
    ada.Text_IO.new_line (datafilex);
    ada.Text_Io.Put (datafilex, "      <UID>");
    for I in 1..2 loop
       UID1 := BMISL.UID1.Random(seed_pkg.G_uid1);
       temp := spec_file_pkg.UID1_type'Image(UID1);
       ada.text_io.put(datafile, temp(2));
       ada.text_io.put(datafilex,temp(2));
    end loop;
    for I in 1..2 loop
       UID2 := BMISL.UID2.Random(seed_pkg.G_uid2);
       BMISL.UID2_IO.put (datafile,  UID2, 5);
       BMISL.UID2_IO.put (datafilex, UID2, 5);
    end loop;
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "</UID>");
    ada.Text_IO.new_line (datafilex);

    ada.Text_Io.put (datafile, "/");

    Trk := BMISL.Trk.Random(seed_pkg.G_Trk);
    BMISL.Trk_IO.put (datafile, Trk, 1);
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "      <TTYP>");
    BMISL.Trk_IO.put (datafilex, Trk, 1);
    ada.Text_Io.Put (datafilex, "</TTYP>");
    ada.Text_IO.new_line (datafilex);

    Share_pkg.Threatcode (datafile, datafilex);

    ada.Text_Io.put (datafile, "////////");

    Event := BMISL.event.Random(seed_pkg.G_event);
    BMISL.Event_IO.put (datafile, Event, 1);
    ada.Text_Io.put (datafile, "/");
    ada.Text_Io.Put (datafilex, "      <RETYP>");
    BMISL.Event_IO.put (datafilex, Event, 1);
    ada.Text_Io.Put (datafilex, "</RETYP>");
    ada.Text_IO.new_line (datafilex);

    Share_pkg.DateTimeGroup (datafile, datafilex);
    ada.Text_Io.Put (datafile, "/");

    Share_pkg.MonthYearField (datafile, datafilex);

    ada.Text_IO.new_line (datafile);
    ada.Text_Io.Put (datafile, "/");  -- denotes continuation from previous line
    ada.Text_Io.Put (datafile, "/");

    Share_pkg.PositionField (datafile, datafilex);

    ada.Text_IO.new_line (datafile);
    ada.Text_Io.Put (datafilex, "</BMISL>");
    ada.Text_IO.new_line (datafilex);

  end Add_BMISL_set;

end BMISL_set_Pkg;

-- **************************
-- Random_files.ada
-- **************************

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
package Msg is new Ada.Numerics.Discrete_Random(spec_file_pkg.Msg_set_type);

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package common is
  package threatcode is new Ada.Numerics.Discrete_Random(spec_file_pkg.Threat_Code_type);
  package day is new Ada.Numerics.Discrete_Random(spec_file_pkg.Day_type);
  package hrs          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Hrs_type);
  package min          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Min_type);
  package tzone        is new Ada.Numerics.Discrete_Random(spec_file_pkg.Tzone_type);
  package cksum        is new Ada.Numerics.Discrete_Random(spec_file_pkg.Cksum_type);
  package month        is new Ada.Numerics.Discrete_Random(spec_file_pkg.Month_type);
  package year         is new Ada.Numerics.Discrete_Random(spec_file_pkg.Year_type);
  package Pos_Lat      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Lat_type);
  package Pos_Lon      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Lon_type);
  package Pos_Dec      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Dec_type);
  package Pos_DirN     is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_DirN_type);
  package Pos_DirE     is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_DirE_type);
  package Pos_LNum     is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_LNum_type);

  package threatcode_IO is new
                             ada.Text_Io.Enumeration_Io(spec_file_pkg.Threat_Code_type);
  package day_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Day_type);
  package hrs_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Hrs_type);
  package min_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Min_type);
  package tzone_IO      is new Ada.Text_Io.Integer_Io(spec_file_pkg.Tzone_type);
  package cksum_IO      is new Ada.Text_Io.Integer_Io(spec_file_pkg.Cksum_type);
  package month_IO      is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Month_type);
  package year_IO       is new Ada.Text_Io.Integer_Io(spec_file_pkg.Year_type);
  package Pos_Lat_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Lat_type);
  package Pos_Lon_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Lon_type);
  package Pos_Dec_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Dec_type);
  package Pos_DirN_IO   is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Pos_DirN_type);
  package Pos_DirE_IO   is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Pos_DirE_type);
  package Pos_LNum_IO   is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_LNum_type);

  end common;


with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package MSGID is
  package Command      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Command_type);
  package Serialnum    is new Ada.Numerics.Discrete_Random(spec_file_pkg.Serialnum_type);
  package Month        is new Ada.Numerics.Discrete_Random(spec_file_pkg.Month_type);
  package Command_IO   is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Command_type);
  package Serialnum_IO is new Ada.Text_Io.Integer_Io(spec_file_pkg.Serialnum_type);
  package Month_IO     is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Month_type);
end MSGID;

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package LCTC is
  package link          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Link_type);
  package tracknum      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Tracknum_type);
  package tracknumA     is new
Ada.Numerics.Discrete_Random(spec_file_pkg.TracknumA_type);
  package report        is new
Ada.Numerics.Discrete_Random(spec_file_pkg.Link_Report_type);
  package exersim       is new
Ada.Numerics.Discrete_Random(spec_file_pkg.Exer_Sim_Ind_type);
  package link_IO       is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Link_type);
  package tracknum_IO   is new Ada.Text_Io.Integer_Io(spec_file_pkg.Tracknum_type);
  package report_IO     is new
Ada.Text_Io.Enumeration_Io(spec_file_pkg.Link_Report_type);
  package exersim_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Exer_Sim_Ind_type);
end LCTC;


with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package BMISL is
  package UID1 is new Ada.Numerics.Discrete_Random(spec_file_pkg.UID1_type);
  package UID2 is new Ada.Numerics.Discrete_Random(spec_file_pkg.UID2_type);
  package Trk  is new Ada.Numerics.Discrete_Random(spec_file_pkg.Trk_type);
  package Event is new Ada.Numerics.Discrete_Random(spec_file_pkg.Event_type);

  package UID2_IO is new Ada.Text_Io.Integer_Io(spec_file_pkg.UID2_type);
  package Trk_IO  is new Ada.Text_Io.Integer_Io(spec_file_pkg.Trk_type);
  package Event_IO is new Ada.Text_Io.Integer_Io(spec_file_pkg.Event_type);

end BMISL;


-- **************************
-- seed_pkg.ada
-- **************************
with Msg;
with common;
with MSGID;
with LCTC;
with BMISL;

package seed_pkg is

  G_Msg      : Msg.generator;

  G_thr      : common.threatcode.generator;
  G_day      : common.day.generator;
  G_hrs      : common.hrs.generator;
  G_min      : common.min.generator;
  G_tzo      : common.tzone.generator;
  G_cks      : common.cksum.generator;
  G_month    : common.month.generator;
  G_year     : common.year.generator;
  G_Pos_Lat  : common.Pos_Lat.generator;
  G_Pos_Lon  : common.Pos_Lon.generator;
  G_Pos_Dec  : common.Pos_Dec.generator;
  G_Pos_DirN : common.Pos_DirN.generator;
  G_Pos_DirE : common.Pos_DirE.generator;
  G_Pos_LNum : common.Pos_LNum.generator;

  G_com : MSGID.Command.generator;
  G_ser : MSGID.Serialnum.generator;
  G_mon : MSGID.Month.generator;

  G_lin : LCTC.link.generator;
  G_tra : LCTC.tracknum.generator;
  G_trB : LCTC.tracknumA.generator;
  G_rep : LCTC.report.generator;
  G_exe : LCTC.exersim.generator;

  G_UID1    : BMISL.UID1.generator;
  G_UID2    : BMISL.UID2.generator;
  G_trk     : BMISL.trk.generator;
  G_event   : BMISL.event.generator;

  procedure init_seed;
end seed_pkg;

package body seed_pkg is
  procedure init_seed is
  begin
    Msg.Reset(G_Msg);

    common.Threatcode.Reset(G_thr);
    common.day.Reset(G_day);
    common.hrs.Reset(G_hrs);
    common.min.Reset(G_min);
    common.tzone.Reset(G_tzo);
    common.cksum.Reset(G_cks);
    common.month.Reset(G_month);
    common.year.Reset(G_year);
    common.Pos_Lat.Reset(G_Pos_lat);
    common.Pos_Lon.Reset(G_Pos_lon);
    common.Pos_Dec.Reset(G_Pos_Dec);
    common.Pos_DirN.Reset(G_Pos_DirN);
    common.Pos_DirE.Reset(G_Pos_DirE);
    common.Pos_LNum.Reset(G_Pos_LNum);

    MSGID.Command.Reset(G_com);
    MSGID.Serialnum.Reset(G_ser);
    MSGID.Month.Reset(G_mon);

    LCTC.link.Reset(G_lin);
    LCTC.tracknum.Reset(G_tra);
    LCTC.tracknumA.Reset(G_trB);
    LCTC.report.Reset(G_rep);
    LCTC.exersim.Reset(G_exe);

    BMISL.UID1.Reset(G_UID1);
    BMISL.UID2.Reset(G_UID2);
    BMISL.trk.Reset(G_trk);
    BMISL.event.Reset(G_event);

  end init_seed;
end seed_pkg;


-- **************************
-- Share_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with common;

package Share_Pkg is
  procedure DateTimeGroup  (datafile:  in out ada.text_io.file_type;
                            datafilex: in out ada.text_io.file_type);
  procedure MonthYearField (datafile:  in out ada.text_io.file_type;
                            datafilex: in out ada.text_io.file_type);
  procedure PositionField  (datafile:  in out ada.text_io.file_type;
                            datafilex: in out ada.text_io.file_type);
  procedure Threatcode     (datafile:  in out ada.text_io.file_type;
                            datafilex: in out ada.text_io.file_type);
end Share_Pkg;

package body Share_Pkg is
  procedure DateTimeGroup (datafile:  in out ada.text_io.file_type;

datafilex: in out ada.text_io.file_type) is
    Day      : spec_file_pkg.Day_type;
    Hrs      : spec_file_pkg.Hrs_type;
    Min      : spec_file_pkg.Min_type;
    Tzone    : spec_file_pkg.Tzone_type;
    Cksum    : spec_file_pkg.Cksum_type;

  begin
    ada.Text_Io.Put (datafilex, "      <DTG>");

    Day := common.day.Random(seed_pkg.G_day);
    common.day_IO.put (datafile,  Day, 2);
    common.day_IO.put (datafilex, Day, 2);

    Hrs := common.hrs.Random(seed_pkg.G_hrs);
    common.hrs_IO.put (datafile,  Hrs, 2);
    common.hrs_IO.put (datafilex, Hrs, 2);

    Min := common.min.Random(seed_pkg.G_min);
    common.min_IO.put (datafile,  Min, 2);
    common.min_IO.put (datafilex, Min, 2);

    ada.Text_Io.put (datafile,  "Z");
    ada.Text_Io.put (datafilex, "Z");
    Tzone := common.tzone.Random(seed_pkg.G_tzo);
    common.tzone_IO.put (datafile,  Tzone, 1);
    common.tzone_IO.put (datafilex, Tzone, 1);

    Cksum := common.cksum.Random(seed_pkg.G_cks);
    common.cksum_IO.put (datafile,  Cksum, 1);
    common.cksum_IO.put (datafilex, Cksum, 1);

    ada.Text_Io.Put (datafilex, "</DTG>");
    ada.Text_IO.new_line (datafilex);
  end DateTimeGroup;

  procedure MonthYearField (datafile:  in out ada.text_io.file_type;
                            datafilex: in out ada.text_io.file_type) is

    Month    : spec_file_pkg.Month_type;
    Year     : spec_file_pkg.Year_type;
    Pos_Lat     : spec_file_pkg.Pos_Lat_type;
    Pos_Lon  : spec_file_pkg.Pos_Lon_type;
    Pos_Dec  : spec_file_pkg.Pos_Dec_type;
    Pos_DirN : spec_file_pkg.Pos_DirN_type;
    Pos_DirE : spec_file_pkg.Pos_DirE_type;
  begin
    ada.Text_Io.Put (datafilex, "     <MOYR>");

    Month := common.month.Random(seed_pkg.G_month);
    common.month_IO.put (datafile,  Month);
    common.month_IO.put (datafilex, Month);

    Year := common.year.Random(seed_pkg.G_year);
    common.year_IO.put (datafile,  Year, 2);
    common.year_IO.put (datafilex, Year, 2);

    ada.Text_Io.Put (datafilex, "</MOYR>");
    ada.Text_IO.new_line (datafilex);
  end MonthYearField;

  procedure PositionField (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type) is
    Pos_Lat     : spec_file_pkg.Pos_Lat_type;
    Pos_Lon  : spec_file_pkg.Pos_Lon_type;
    Pos_Dec  : spec_file_pkg.Pos_Dec_type;
    Pos_DirN : spec_file_pkg.Pos_DirN_type;
    Pos_DirE : spec_file_pkg.Pos_DirE_type;
    Pos_LNum : spec_file_pkg.Pos_LNum_type;
  begin
    ada.Text_Io.Put (datafilex, "     <POSIT>");

    ada.Text_Io.put (datafile,  "LL:");
    ada.Text_Io.put (datafilex, "LL:");

    Pos_Lat := common.pos_lat.Random(seed_pkg.G_Pos_lat);
    common.pos_lat_IO.put (datafile,  Pos_Lat, 2);
    common.pos_lat_IO.put (datafilex, Pos_Lat, 2);

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 4);
    common.pos_dec_IO.put (datafilex, Pos_dec, 4);

    Pos_DirN := common.pos_dirN.Random(seed_pkg.G_Pos_dirN);
    common.pos_dirN_IO.put (datafile,  Pos_dirN);
    common.pos_dirN_IO.put (datafilex, Pos_dirN);

    Pos_LNum := common.pos_LNum.Random(seed_pkg.G_Pos_LNum);
    common.pos_LNum_IO.put (datafile,  Pos_LNum, 1);
    common.pos_LNum_IO.put (datafilex, Pos_LNum, 1);

    ada.Text_Io.put (datafile,  "-");
    ada.Text_Io.put (datafilex, "-");

    Pos_Lon := common.pos_lon.Random(seed_pkg.G_Pos_lon);
```

```
    common.pos_lon_IO.put (datafile,  Pos_Lon, 3);
    common.pos_lon_IO.put (datafilex, Pos_Lon, 3);

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 4);
    common.pos_dec_IO.put (datafilex, Pos_dec, 4);

    Pos_DirE := common.pos_dirE.Random(seed_pkg.G_Pos_dirE);
    common.pos_dirE_IO.put (datafile,  Pos_dirE);
    common.pos_dirE_IO.put (datafilex, Pos_dirE);

    Pos_LNum := common.pos_LNum.Random(seed_pkg.G_Pos_LNum);
    common.pos_LNum_IO.put (datafile,  Pos_LNum, 1);
    common.pos_LNum_IO.put (datafilex, Pos_LNum, 1);

    ada.Text_Io.Put (datafilex, "</POSIT>");
    ada.Text_IO.new_line (datafilex);
  end PositionField;

  procedure Threatcode (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type) is
    Threat_Code : spec_file_pkg.Threat_Code_type;
  begin
    Threat_Code := common.threatcode.Random(seed_pkg.G_thr);
    common.threatcode_IO.put (datafile, Threat_Code);
    ada.Text_Io.Put (datafilex, "      <THREAT>");
    common.threatcode_IO.put (datafilex, Threat_Code);
    ada.Text_Io.Put (datafilex, "/THREAT>");
    ada.Text_IO.new_line (datafilex);
  end Threatcode;

end Share_Pkg;


-- **************************
-- spec_file_pkg
-- **************************
package spec_file_pkg is

  type Msg_set_type is (BLink_TMS,
                        XLink_TMS,
                        TBM_TMS);
--                      COP_SMS,
--                      COP_UOMS,
--                      DbSync_MS);

  type Threat_Code_type is (PND,
                            UNK,
                            FRD,
                            AFD,
                            NEU,
                            SUS,
                            HOS);
  subtype Day_type      is Integer Range 10..31;
  subtype Hrs_type      is Integer Range 10..23;
  subtype Min_type      is Integer Range 10..59;
  subtype Tzone_type    is Integer Range 1..9;
  subtype Cksum_type    is Integer Range 0..9;
  subtype Year_Type     is Integer Range 90..99;
  subtype Pos_Lat_type  is Integer Range 10..89;
  subtype Pos_Lon_type  is Integer Range 100..179;
  subtype Pos_Dec_type  is Integer Range 1000..9999;
  subtype Pos_LNum_type is Integer Range 1..9;
  type Pos_DirN_type    is (N, S);
  type Pos_DirE_type    is (E, W);

  type Command_type is (CTF70,
                        COMTHIRDFLT,
                        CTG810,
                        KITTYHAWK,
                        FTHOOD,
                         FTKNOX,
                         TACOM,
                         TARDEC,
                         NPS);
  subtype Serialnum_type is Integer range 1000 .. 9999;
  type Month_type is (JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC);

  type Link_type is (L11,L11B,L16,L4A,L22);
  subtype Tracknum_type is Integer range 1000 .. 9999;
  subtype TracknumA_type is Character range 'A' .. 'Z';
  type Link_Report_type is (AIR,
                             SUB,
                             GROUND,
                             UNKNOWN,
                             SPACE,
                             ALIEN,
                             OTHER);
  subtype Exer_Sim_Ind_type is Integer range 0 .. 1;

  subtype UID1_type is Character Range 'A'..'Z';
  subtype UID2_type is Integer Range 10000..99999;
  subtype trk_type is Integer Range 1..3;
  subtype Event_type is Integer Range 1..5;

end spec_file_pkg;
```

# APPENDIX K: SOURCE LISTING FOR USMTF FORMATTED MESSAGE

```
-- *******************
-- Main.ada
-- *******************

with ada.text_io;
with Basic_Msg_Pkg;

procedure main_mtf is
  datafile : ada.text_io.file_type;
  datafilex: ada.text_io.file_type;

begin
  Basic_Msg_Pkg.Initialize (datafile, datafilex);
  Basic_Msg_Pkg.Generate_TestMessages (datafile,datafilex);
  Basic_Msg_Pkg.Closefile (datafile,datafilex);
end main_mtf;


-- ******************************
-- Basic_Msg_Pkg.ada
-- ******************************

with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with Msg;
with EXER_set_pkg;
with MSGID_set_pkg;
with WEATHLOC_set_pkg;
with OBSTIME_set_pkg;
with WIND_set_pkg;
with SuperAIR_set_pkg;

package Basic_Msg_Pkg is
  procedure Initialize (datafile:  out ada.text_io.file_type;
                        datafilex: out ada.text_io.file_type);
  procedure Add_Header (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type);
  procedure Add_Footer (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type);
  procedure Closefile  (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type);
  procedure Generate_TestMessages (datafile:  in out ada.text_io.file_type;
                                   datafilex: in out ada.text_io.file_type);
  procedure Generate_Messages (datafile:  in out ada.text_io.file_type;
                               datafilex: in out ada.text_io.file_type);
end Basic_Msg_Pkg;

package body Basic_Msg_Pkg is

  procedure Initialize (datafile:  out ada.text_io.file_type;
                        datafilex: out ada.text_io.file_type) is
  begin
    seed_pkg.init_seed;
    ada.Text_Io.Create (File=> datafile,
                        Mode=> Ada.Text_Io.out_file,
                        Name=> "A:\Msg_File_MTF_nat.txt");
    ada.Text_Io.Create (File=> datafilex,
                        Mode=> Ada.Text_Io.out_file,
                        Name=> "A:\Msg_File_MTF_XML.txt");
  end Initialize;

  procedure Add_Header (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put_Line (datafile,  "BT");
    ada.Text_Io.Put_Line (datafilex, "BT");
  end Add_Header;

  procedure Add_Footer (datafile:  in out ada.text_io.file_type;
                        datafilex: in out ada.text_io.file_type) is
  begin
    ada.Text_Io.Put_Line (datafile,  "BT");
    ada.Text_Io.Put_Line (datafilex, "BT");
    for i in 1..7 loop
      ada.text_io.New_Line (datafile);
      ada.text_io.New_Line (datafilex);
    end loop;
    ada.text_io.Put_Line (datafile,  "NNNN");
    ada.text_io.Put_Line (datafilex, "NNNN");
  end Add_Footer;

  procedure Closefile (datafile:  in out ada.text_io.file_type;
                       datafilex: in out ada.text_io.file_type) is
  begin
    ada.text_io.close (datafile);
    ada.text_io.close (datafilex);
  end Closefile;

  procedure Generate_TestMessages (datafile:  in out ada.text_io.file_type;
                                   datafilex: in out ada.text_io.file_type) is
  begin
    for I in 1 .. 20 loop
      Add_Header (datafile, datafilex);
      EXER_set_Pkg.Add_EXER_set   (datafile, datafilex);
      MSGID_set_Pkg.Add_MSGID_set (datafile, datafilex);
      WEATHLOC_set_Pkg.Add_WeatherLoc_set (datafile, datafilex);
      OBSTIME_set_Pkg.Add_ObservationTime_set (datafile, datafilex);
      WIND_set_Pkg.Add_Wind_set (datafile, datafilex);
      SuperAIR_set_Pkg.Add_Superair_set (datafile, datafilex);
      Add_Footer (datafile, datafilex);
    end loop;
    end Generate_TestMessages;

  procedure Generate_Messages (datafile:  in out ada.text_io.file_type;
                               datafilex: in out ada.text_io.file_type) is
  Msg_set : spec_file_pkg.Msg_set_type;
  begin
    for I in 1..20 loop
      Msg_set := Msg.Random(seed_pkg.G_msg);
               Add_Header (datafile, datafilex);
      case Msg_set is
        when spec_file_pkg.WXOBS =>
          EXER_set_Pkg.Add_EXER_set (datafile, datafilex);
          MSGID_set_Pkg.Add_MSGID_set (datafile, datafilex);
          WEATHLOC_set_Pkg.Add_WeatherLoc_set (datafile, datafilex);
          OBSTIME_set_Pkg.Add_ObservationTime_set (datafile, datafilex);
          WIND_set_Pkg.Add_Wind_set (datafile, datafilex);
        when others =>
          null;
      end case;
      Add_Footer (datafile, datafilex);
    end loop;
  end Generate_Messages;

end Basic_Msg_Pkg;


-- **************************
-- MSGID_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with MSGID;

package MSGID_set_Pkg is
```

```
  procedure Add_MSGID_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type);
end MSGID_set_Pkg;

package body MSGID_set_Pkg is

  procedure Add_MSGID_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type) is
    Command    : spec_file_pkg.Command_type;
    Originator : spec_file_pkg.Originator_type;
    Month                : spec_file_pkg.Month_type;

  begin
    ada.Text_Io.Put (datafile,  "MSGID/");
    ada.Text_Io.Put_line (datafilex, "<message_identification>");

    Command := MSGID.Command.Random(seed_pkg.G_com);
    MSGID.Command_IO.put (datafile,  Command);
    ada.Text_Io.put (datafile,  "/");
    ada.Text_Io.Put (datafilex, "  <message_text_format_identifier>");
    MSGID.Command_IO.put (datafilex, Command);
    ada.Text_Io.Put_line (datafilex, "</message_text_format_identifier>");

    Originator := MSGID.Originator.Random(seed_pkg.G_org);
    MSGID.Originator_IO.put (datafile,  Originator);
    ada.Text_Io.Put (datafilex, "  <originator>");
    MSGID.Originator_IO.put (datafilex, Originator);
    ada.Text_Io.Put_line (datafilex, "</originator>");

    ada.Text_Io.put_line (datafile,  "//");
    ada.Text_Io.Put_line (datafilex, "</message_identification>");

  end Add_MSGID_set;

end MSGID_set_Pkg;



-- **************************
-- OBSTIME_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with common;

package OBSTIME_set_Pkg is
  procedure Add_ObservationTime_set (datafile:  in out ada.text_io.file_type;
                                     datafilex: in out ada.text_io.file_type);
end OBSTIME_set_Pkg;

package body OBSTIME_set_Pkg is
  procedure Add_ObservationTime_set (datafile:  in out ada.text_io.file_type;
                                     datafilex: in out ada.text_io.file_type) is
    Day      : spec_file_pkg.Day_type;
    Hrs      : spec_file_pkg.Hrs_type;
    Min      : spec_file_pkg.Min_type;
  begin
    ada.Text_Io.Put (datafile, "OBSTIME/");
    ada.Text_Io.Put_line (datafilex, "<observation_day_time>");

    Day := common.day.Random(seed_pkg.G_day);
    common.day_IO.put (datafile,  Day, 2);
    ada.Text_Io.Put (datafilex, "  <day>");
    common.day_IO.put (datafilex, Day, 2);
    ada.Text_Io.Put_line (datafilex, "</day>");

    Hrs := common.hrs.Random(seed_pkg.G_hrs);
    common.hrs_IO.put (datafile,  Hrs, 2);
    ada.Text_Io.Put (datafilex, "  <time_hour>");
    common.hrs_IO.put (datafilex, Hrs, 2);
    ada.Text_Io.Put_line (datafilex, "</time_hour>");

    Min := common.min.Random(seed_pkg.G_min);
    common.min_IO.put (datafile,  Min, 2);
    ada.Text_Io.Put (datafilex, "  <time_min>");
    common.min_IO.put (datafilex, Min, 2);
    ada.Text_Io.Put_line (datafilex, "</time_min>");

    ada.Text_Io.put (datafile,  "Z");
    ada.Text_Io.Put (datafilex, "  <time_zone>");
    ada.Text_Io.put (datafilex, "Z");
    ada.Text_Io.Put_line (datafilex, "</time_zone>");

    ada.Text_Io.Put_line (datafilex, "</observation_day_time>");
    ada.Text_IO.Put_line (datafile, "//");

  end Add_ObservationTime_set;
end OBSTIME_set_Pkg;



-- **************************
-- WXOBS_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with share_pkg;

package WXOBS_set_Pkg is
  procedure Add_WXOBS_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type);
end WXOBS_set_Pkg;

package body WXOBS_set_Pkg is

  procedure Add_WXOBS_set (datafile:  in out ada.text_io.file_type;
                           datafilex: in out ada.text_io.file_type) is

  begin
    share_pkg.Weatherloc (datafile, datafilex);
    share_pkg.obstime (datafile, datafilex);
    share_pkg.wind (datafile, datafilex);
  end Add_WXOBS_set;

end WXOBS_set_Pkg;



-- **************************
-- WIND_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with Wind;

package WIND_set_Pkg is
  procedure Add_Wind_set (datafile:  in out ada.text_io.file_type;

       datafilex: in out ada.text_io.file_type);
end WIND_set_Pkg;

package body WIND_set_Pkg is
  procedure Add_Wind_set (datafile:  in out ada.text_io.file_type;
                          datafilex: in out ada.text_io.file_type) is
  Wind_dir   : spec_file_pkg.Wind_dir_type;
  Wind_speed : spec_file_pkg.Wind_speed_type;
  begin
    ada.text_Io.Put (datafile, "WIND/");
    ada.Text_Io.Put_line (datafilex, "<wind>");

    Wind_dir := Wind.wind_dir.Random(seed_pkg.G_Wind_dir);
    Wind.wind_dir_Io.put (datafile, wind_dir, 3);
    ada.Text_Io.Put (datafilex, "  <wind_direction>");
    Wind.wind_dir_Io.put (datafilex, wind_dir, 3);
    ada.Text_Io.Put_line (datafilex, "</wind_direction>");

    ada.text_Io.Put (datafile, "/-/");
    ada.Text_Io.Put_line (datafilex, "  <variable_wind_direction/>");

    Wind_speed := Wind.wind_speed.Random(seed_pkg.G_Wind_speed);
    Wind.wind_speed_Io.put (datafile, wind_speed, 2);
    ada.Text_Io.Put (datafilex, "  <wind_speed>");
    Wind.wind_speed_Io.put (datafilex, wind_speed, 2);
    ada.Text_Io.Put_line (datafilex, "</wind_speed>");

    ada.text_Io.Put_line (datafile, "//");
    ada.Text_Io.Put_line (datafilex, "</wind>");
  end Add_Wind_set;
end WIND_set_Pkg;



-- **************************
-- WEATHLOC_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with common;

package WEATHLOC_set_Pkg is
  procedure Add_WeatherLoc_set (datafile:  in out ada.text_io.file_type;
                                datafilex: in out ada.text_io.file_type);
end WEATHLOC_set_Pkg;

package body WEATHLOC_set_Pkg is
  procedure Add_WeatherLoc_set (datafile:  in out ada.text_io.file_type;
                                datafilex: in out ada.text_io.file_type) is
    Pos_Lat     : spec_file_pkg.Pos_Lat_type;
    Pos_Lon  : spec_file_pkg.Pos_Lon_type;
    Pos_Dec  : spec_file_pkg.Pos_Dec_type;
    Pos_DirN : spec_file_pkg.Pos_DirN_type;
    Pos_DirE : spec_file_pkg.Pos_DirE_type;
  begin
    ada.text_Io.Put (datafile, "WEATHLOC/");
    ada.Text_Io.Put_line (datafilex, "<location_of_weather>");

    Pos_Lat := common.pos_lat.Random(seed_pkg.G_Pos_lat);
    common.pos_lat_IO.put (datafile,  Pos_Lat, 2);
    ada.Text_Io.Put (datafilex, "  <latitude_degrees>");
    common.pos_lat_IO.put (datafilex, Pos_Lat, 2);
    ada.Text_Io.Put_line (datafilex, "</latitude_degrees>");

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 2);
    ada.Text_Io.Put (datafilex, "  <latitude_minutes>");
    common.pos_dec_IO.put (datafilex, Pos_dec, 2);
    ada.Text_Io.Put_line (datafilex, "</latitude_minutes>");

    Pos_DirN := common.pos_dirN.Random(seed_pkg.G_Pos_dirN);
    common.pos_dirN_IO.put (datafile,  Pos_dirN);
    ada.Text_Io.Put (datafilex, "  <latitude_hemisphere>");
    common.pos_dirN_IO.put (datafilex, Pos_dirN);
    ada.Text_Io.Put_line (datafilex, "</latitude_hemisphere>");

    Pos_Lon := common.pos_lon.Random(seed_pkg.G_Pos_lon);
    common.pos_lon_IO.put (datafile,  Pos_Lon, 3);
    ada.Text_Io.Put (datafilex, "  <longitude_degrees>");
    common.pos_lon_IO.put (datafilex, Pos_Lon, 3);
    ada.Text_Io.Put_line (datafilex, "</longitude_degrees>");

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 2);
    ada.Text_Io.Put (datafilex, "  <longitude_minutes>");
    common.pos_dec_IO.put (datafilex, Pos_dec, 2);
    ada.Text_Io.Put_line (datafilex, "</longitude_minutes>");

    Pos_DirE := common.pos_dirE.Random(seed_pkg.G_Pos_dirE);
    common.pos_dirE_IO.put (datafile,  Pos_dirE);
    ada.Text_Io.Put (datafilex, "  <longitude_hemisphere>");
    common.pos_dirE_IO.put (datafilex, Pos_dirE);
    ada.Text_Io.Put_line (datafilex, "</longitude_hemisphere>");

    ada.Text_Io.Put_line (datafilex, "</location_of_weather>");
    ada.Text_IO.Put_line (datafile, "//");
  end Add_WeatherLoc_set;
end WEATHLOC_set_Pkg;



-- **************************
-- OBSTIME_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with common;

package OBSTIME_set_Pkg is
  procedure Add_ObservationTime_set (datafile:  in out ada.text_io.file_type;
                                     datafilex: in out ada.text_io.file_type);
end OBSTIME_set_Pkg;

package body OBSTIME_set_Pkg is
  procedure Add_ObservationTime_set (datafile:  in out ada.text_io.file_type;
                                     datafilex: in out ada.text_io.file_type) is
    Day      : spec_file_pkg.Day_type;
    Hrs      : spec_file_pkg.Hrs_type;
    Min      : spec_file_pkg.Min_type;
  begin
    ada.Text_Io.Put (datafile, "OBSTIME/");
    ada.Text_Io.Put_line (datafilex, "<observation_day_time>");

    Day := common.day.Random(seed_pkg.G_day);
    common.day_IO.put (datafile,  Day, 2);
    ada.Text_Io.Put (datafilex, "  <day>");
    common.day_IO.put (datafilex, Day, 2);
    ada.Text_Io.Put_line (datafilex, "</day>");

    Hrs := common.hrs.Random(seed_pkg.G_hrs);
    common.hrs_IO.put (datafile,  Hrs, 2);
    ada.Text_Io.Put (datafilex, "  <time_hour>");
    common.hrs_IO.put (datafilex, Hrs, 2);
    ada.Text_Io.Put_line (datafilex, "</time_hour>");

    Min := common.min.Random(seed_pkg.G_min);
    common.min_IO.put (datafile,  Min, 2);
    ada.Text_Io.Put (datafilex, "  <time_min>");
    common.min_IO.put (datafilex, Min, 2);
    ada.Text_Io.Put_line (datafilex, "</time_min>");

    ada.Text_Io.put (datafile,  "Z");
    ada.Text_Io.Put (datafilex, "  <time_zone>");
    ada.Text_Io.put (datafilex, "Z");
    ada.Text_Io.Put_line (datafilex, "</time_zone>");
    ada.Text_Io.Put_line (datafilex, "</observation_day_time>");
    ada.Text_IO.Put_line (datafile, "//");

  end Add_ObservationTime_set;
end OBSTIME_set_Pkg;



-- **************************
-- SuperAIR_set_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with Wind;

package SuperAIR_set_Pkg is
  procedure Add_SuperAir_set (datafile:  in out ada.text_io.file_type;
                              datafilex: in out ada.text_io.file_type);
end SuperAIR_set_Pkg;

package body SuperAIR_set_Pkg is
  procedure Add_SuperAir_set (datafile:  in out ada.text_io.file_type;
                              datafilex: in out ada.text_io.file_type) is
    SuperAir_Temp : spec_file_pkg.SuperAir_Temp_type;
    Wind_dir      : spec_file_pkg.Wind_dir_type;
    Wind_speed    : spec_file_pkg.Wind_speed_type;
  begin
    ada.text_Io.Put_Line (datafile, "5UPERAIR");
    ada.Text_Io.Put_line (datafilex, "<5uperair_data>");
    ada.text_Io.Put_Line (datafile, "/ALT    /TMPC/DEWTEMP/WINDIR/SPD");

    for i in 1..2 loop
      if i=1 then
        ada.text_Io.Put(datafile, "/M5KFT  /");
      else
        ada.text_io.new_line(datafile);
        ada.text_Io.Put(datafile, "/M10KFT /");
      end if;
      SuperAir_Temp := Wind.SuperAir_Temp.Random(seed_pkg.G_SuperAir_Temp);
      Wind.SuperAir_Temp_Io.put (datafile, SuperAir_Temp, 4);
      ada.text_Io.Put (datafile, "/");
      ada.integer_text_io.put (datafile, integer(SuperAir_Temp +5), 7);
      ada.text_Io.Put (datafile, "/");
      Wind_dir := Wind.wind_dir.Random(seed_pkg.G_Wind_dir);
      Wind.wind_dir_Io.put (datafile, wind_dir, 6);
      ada.text_Io.Put (datafile, "/");
      Wind_speed := Wind.wind_speed.Random(seed_pkg.G_Wind_speed);
      Wind.wind_speed_Io.put (datafile, wind_speed, 4);
      ada.text_Io.Put (datafile, "/");
      ada.Text_Io.Put_line (datafilex, "  <5uperair_data_group_of_fields>");
      ada.Text_Io.Put (datafilex, "    <temperature_celsius>");
      Wind.SuperAir_Temp_Io.put (datafilex, SuperAir_Temp, 3);
      ada.Text_Io.Put_line (datafilex, "</temperature_celsius>");
      ada.Text_Io.Put (datafilex, "    <dewpoint_temperature_celsius>");
      ada.integer_text_io.put (datafilex, integer(SuperAir_Temp +5), 3);
      ada.Text_Io.Put_line (datafilex, "</dewpoint_temperature_celsius>");
      ada.Text_Io.Put (datafilex, "    <wind_direction>");
      Wind.wind_dir_Io.put (datafilex, wind_dir, 3);
      ada.Text_Io.Put_line (datafilex, "</wind_direction>");
      ada.Text_Io.Put (datafilex, "    <peak_gusts_knots>");
      Wind.wind_speed_Io.put (datafilex, wind_speed, 2);
      ada.Text_Io.Put_line (datafilex, "</peak_gusts_knots>");
      ada.Text_Io.Put_line (datafilex, "  </5uperair_data_group_of_fields>");
    end loop;
    ada.text_Io.Put_line (datafile, "/");
    ada.Text_Io.Put_line (datafilex, "<5uperair_data>");
    end Add_SuperAir_set;
end SuperAIR_set_Pkg;
```

```
-- **************************
-- Random_files.ada
-- **************************

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
package Msg is new Ada.Numerics.Discrete_Random(spec_file_pkg.Msg_set_type);

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package common is
  package day          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Day_type);
  package hrs          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Hrs_type);
  package min          is new Ada.Numerics.Discrete_Random(spec_file_pkg.Min_type);
  package month        is new Ada.Numerics.Discrete_Random(spec_file_pkg.Month_type);
  package year         is new Ada.Numerics.Discrete_Random(spec_file_pkg.Year_type);
  package Pos_Lat      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Lat_type);
  package Pos_Lon      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Lon_type);
  package Pos_Dec      is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_Dec_type);
  package Pos_DirN     is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_DirN_type);
  package Pos_DirE     is new Ada.Numerics.Discrete_Random(spec_file_pkg.Pos_DirE_type);

  package day_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Day_type);
  package hrs_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Hrs_type);
  package min_IO        is new Ada.Text_Io.Integer_Io(spec_file_pkg.Min_type);
  package month_IO      is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Month_type);
  package year_IO       is new Ada.Text_Io.Integer_Io(spec_file_pkg.Year_type);
  package Pos_Lat_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Lat_type);
  package Pos_Lon_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Lon_type);
  package Pos_Dec_IO    is new Ada.Text_Io.Integer_Io(spec_file_pkg.Pos_Dec_type);
  package Pos_DirN_IO   is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Pos_DirN_type);
  package Pos_DirE_IO   is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Pos_DirE_type);
end common;

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package EXER is
  package Exercise is new Ada.Numerics.Discrete_Random(spec_file_pkg.Exercise_type);
  package Exercise_IO is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Exercise_type);
end EXER;

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package MSGID is
  package Command    is new Ada.Numerics.Discrete_Random(spec_file_pkg.Command_type);
  package Originator is new Ada.Numerics.Discrete_Random(spec_file_pkg.Originator_type);
  package Command_IO is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Command_type);
  package Originator_IO is new Ada.Text_Io.Enumeration_Io(spec_file_pkg.Originator_type);
end MSGID;

with Ada.Numerics.Discrete_Random;
with spec_file_pkg;
with Ada.Text_Io;
package Wind is
  package wind_dir    is new Ada.Numerics.Discrete_Random(spec_file_pkg.wind_dir_type);
  package wind_speed  is new Ada.Numerics.Discrete_Random(spec_file_pkg.wind_speed_type);
  package Superair_temp is new
Ada.Numerics.Discrete_Random(spec_file_pkg.Superair_temp_type);
  package wind_dir_IO is new Ada.Text_Io.Integer_Io(spec_file_pkg.wind_dir_type);
  package wind_speed_IO is new Ada.Text_Io.Integer_Io(spec_file_pkg.wind_speed_type);
  package Superair_temp_IO is new
Ada.Text_Io.Integer_Io(spec_file_pkg.Superair_temp_type);
end Wind;

-- **************************
-- seed_pkg.ada
-- **************************
with Msg;
with common;
with EXER;
with MSGID;
with Wind;

package seed_pkg is

  G_Msg : Msg.generator;

  G_day      : common.day.generator;
  G_hrs      : common.hrs.generator;
  G_min      : common.min.generator;
  G_month    : common.month.generator;
  G_year     : common.year.generator;
  G_Pos_Lat  : common.Pos_Lat.generator;
  G_Pos_Lon  : common.Pos_Lon.generator;
  G_Pos_Dec  : common.Pos_Dec.generator;
  G_Pos_DirN : common.Pos_DirN.generator;
  G_Pos_DirE : common.Pos_DirE.generator;

  G_exr : EXER.Exercise.generator;

  G_com : MSGID.Command.generator;
  G_org : MSGID.Originator.generator;

  G_wind_dir   : Wind.wind_dir.generator;
  G_wind_speed : Wind.wind_speed.generator;
  G_SuperAir_Temp : Wind.Superair_temp.generator;

  procedure init_seed;
end seed_pkg;

package body seed_pkg is
  procedure init_seed is
  begin
    Msg.Reset(G_Msg);

    common.day.Reset(G_day);
    common.hrs.Reset(G_hrs);
    common.min.Reset(G_min);
    common.month.Reset(G_month);
    common.year.Reset(G_year);
    common.Pos_Lat.Reset(G_Pos_lat);
    common.Pos_Lon.Reset(G_Pos_lon);
    common.Pos_Dec.Reset(G_Pos_Dec);
    common.Pos_DirN.Reset(G_Pos_DirN);
    common.Pos_DirE.Reset(G_Pos_DirE);

    EXER.Exercise.Reset(G_Exr);

    MSGID.Command.Reset(G_com);
    MSGID.Originator.Reset(G_org);

    Wind.wind_dir.Reset(G_wind_dir);
    Wind.wind_speed.Reset(G_wind_speed);
    Wind.Superair_temp.Reset(G_Superair_temp);


  end init_seed;
end seed_pkg;


-- **************************
-- Share_pkg.ada
-- **************************
with ada.text_io;
with ada.integer_text_io;
with seed_pkg;
with spec_file_pkg;
with common;
with WXOBS;

package Share_Pkg is
  procedure ObsTime        (datafile:  in out ada.text_io.file_type;
                             datafilex: in out ada.text_io.file_type);
  procedure Wind           (datafile:  in out ada.text_io.file_type;
                             datafilex: in out ada.text_io.file_type);

  procedure WeatherLoc     (datafile:  in out ada.text_io.file_type;
                             datafilex: in out ada.text_io.file_type);
end Share_Pkg;

package body Share_Pkg is

  procedure ObsTime (datafile:  in out ada.text_io.file_type;
                     datafilex: in out ada.text_io.file_type) is
    Day      : spec_file_pkg.Day_type;
    Hrs      : spec_file_pkg.Hrs_type;
    Min      : spec_file_pkg.Min_type;

  begin
    ada.Text_Io.Put (datafile, "OBSTIME/");
    ada.Text_Io.Put_line (datafilex, "<observation_day_time>");

    Day := common.day.Random(seed_pkg.G_day);
    common.day_IO.put (datafile,  Day, 2);
    ada.Text_Io.Put (datafilex, "  <day>");
    common.day_IO.put (datafilex, Day, 2);
    ada.Text_Io.Put_line (datafilex, "</day>");

    Hrs := common.hrs.Random(seed_pkg.G_hrs);
    common.hrs_IO.put (datafile,  Hrs, 2);
    ada.Text_Io.Put (datafilex, "  <time_hour>");
    common.hrs_IO.put (datafilex, Hrs, 2);
    ada.Text_Io.Put_line (datafilex, "</time_hour>");

    Min := common.min.Random(seed_pkg.G_min);
    common.min_IO.put (datafile,  Min, 2);
    ada.Text_Io.Put (datafilex, "  <time_min>");
    common.min_IO.put (datafilex, Min, 2);
    ada.Text_Io.Put_line (datafilex, "</time_min>");

    ada.Text_Io.put (datafile,  "Z");
    ada.Text_Io.Put (datafilex, "  <time_zone>");
    ada.Text_Io.put (datafilex, "Z");
    ada.Text_Io.Put_line (datafilex, "</time_zone>");

    ada.Text_Io.Put_line (datafilex, "</observation_day_time>");
    ada.Text_IO.Put_line (datafile, "//");
  end Obstime;

  procedure Wind (datafile:  in out ada.text_io.file_type;
                  datafilex: in out ada.text_io.file_type) is
    Wind_dir   : spec_file_pkg.Wind_dir_type;
    Wind_speed : spec_file_pkg.Wind_speed_type;
  begin
    ada.text_Io.Put (datafile, "WIND/");
    ada.Text_Io.Put_line (datafilex, "<wind>");

    Wind_dir := WXOBS.wind_dir.Random(seed_pkg.G_Wind_dir);
    WXOBS.wind_dir_Io.put (datafile, wind_dir, 3);
    ada.Text_Io.Put (datafilex, "  <wind_direction>");
    WXOBS.wind_dir_Io.put (datafilex, wind_dir, 3);
    ada.Text_Io.Put_line (datafilex, "</wind_direction>");

    ada.text_Io.Put (datafile, "/-/");
    ada.Text_Io.Put_line (datafilex, "  <variable_wind_direction/>");

    Wind_speed := WXOBS.wind_speed.Random(seed_pkg.G_Wind_speed);
    WXOBS.wind_speed_Io.put (datafile, wind_speed, 2);
    ada.Text_Io.Put (datafilex, "  <wind_speed>");
    WXOBS.wind_speed_Io.put (datafilex, wind_speed, 2);
    ada.Text_Io.Put_line (datafilex, "</wind_speed>");

    ada.text_Io.Put (datafile, "//");
    ada.Text_Io.Put_line (datafilex, "</wind>");
  end wind;


  procedure WeatherLoc (datafile:  in out ada.text_io.file_type;
                       datafilex: in out ada.text_io.file_type) is
    Pos_Lat     : spec_file_pkg.Pos_Lat_type;
    Pos_Lon  : spec_file_pkg.Pos_Lon_type;
    Pos_Dec  : spec_file_pkg.Pos_Dec_type;
    Pos_DirN : spec_file_pkg.Pos_DirN_type;
    Pos_DirE : spec_file_pkg.Pos_DirE_type;
  begin
    ada.text_Io.Put (datafile, "WEATHLOC/");
    ada.Text_Io.Put_line (datafilex, "<location_of_weather>");

    Pos_Lat := common.pos_lat.Random(seed_pkg.G_Pos_lat);
    common.pos_lat_IO.put (datafile,  Pos_Lat, 2);
    ada.Text_Io.Put (datafilex, "  <latitude_degrees>");
    common.pos_lat_IO.put (datafilex, Pos_Lat, 2);
    ada.Text_Io.Put_line (datafilex, "</latitude_degrees>");

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 2);
    ada.Text_Io.Put (datafilex, "  <latitude_minutes>");
    common.pos_dec_IO.put (datafilex, Pos_dec, 2);
    ada.Text_Io.Put_line (datafilex, "</latitude_minutes>");

    Pos_DirN := common.pos_dirN.Random(seed_pkg.G_Pos_dirN);
    common.pos_dirN_IO.put (datafile,  Pos_dirN);
    ada.Text_Io.Put (datafilex, "  <latitude_hemisphere>");
    common.pos_dirN_IO.put (datafilex, Pos_dirN);
    ada.Text_Io.Put_line (datafilex, "</latitude_hemisphere>");

    Pos_Lon := common.pos_lon.Random(seed_pkg.G_Pos_lon);
    common.pos_lon_IO.put (datafile,  Pos_Lon, 3);
    ada.Text_Io.Put (datafilex, "  <longitude_degrees>");
    common.pos_lon_IO.put (datafilex, Pos_Lon, 3);
    ada.Text_Io.Put_line (datafilex, "</longitude_degrees>");

    Pos_Dec := common.pos_dec.Random(seed_pkg.G_Pos_dec);
    common.pos_dec_IO.put (datafile,  Pos_dec, 2);
    ada.Text_Io.Put (datafilex, "  <longitude_minutes>");
    common.pos_dec_IO.put (datafilex, Pos_dec, 2);
    ada.Text_Io.Put_line (datafilex, "</longitude_minutes>");

    Pos_DirE := common.pos_dirE.Random(seed_pkg.G_Pos_dirE);
    common.pos_dirE_IO.put (datafile,  Pos_dirE);
    ada.Text_Io.Put (datafilex, "  <longitude_hemisphere>");
    common.pos_dirE_IO.put (datafilex, Pos_dirE);
    ada.Text_Io.Put_line (datafilex, "</longitude_hemisphere>");

    ada.Text_Io.Put_line (datafilex, "</location_of_weather>");
    ada.Text_IO.Put_line (datafile, "//");
       end Weatherloc;


end Share_Pkg;

-- **************************
-- spec_file_pkg
-- **************************
package spec_file_pkg is

  type Msg_set_type is (NBC1,
                        WXOBS,
                        TARGET);
```

```
        type Exercise_type is (DESERT_STORM,
                               DESERT_WIND,
                               OLIVE_DRAB,
                               MISSION_IMPOSSIBLE,
                               EAGLE_HUNT,
                               RED_OCTOBER,
                               KOLE_SEARCH,
                               MAHI_MAHI);

   subtype Day_type      is Integer Range 10..31;
   subtype Hrs_type      is Integer Range 10..23;
   subtype Min_type      is Integer Range 10..59;
   subtype Year_Type     is Integer Range 90..99;
   subtype Pos_Lat_type  is Integer Range 10..89;
   subtype Pos_Lon_type  is Integer Range 100..179;
   subtype Pos_Dec_type  is Integer Range 10..59;
   type Pos_DirN_type    is (N, S);
   type Pos_DirE_type    is (E, W);

   subtype Wind_dir_type is Integer Range 100..359;
   subtype Wind_speed_type is Integer Range 10..99;
        subtype SuperAir_Temp_type is Integer Range -50..100;

   type Command_type is (NBC1,
                         NBC2,
                         NBC3,
                         WXOBS,
                         TARGET);
   type Originator_type is (CTF70,
                            COMTHIRDFLT,
                            CTG810,
                            KITTYHAWK,
                            FTHOOD,
                            FTKNOX,
                            TACOM,
                            TARDEC,
                            NPS);

   type Month_type is (JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC);

end spec_file_pkg;
```

## APPENDIX L: SAMPLE RUN TIME LIST FOR CIX MESSAGE

### 1. MESSAGES GENERATED IN NATIVE FORMAT:

```
BT
MSGID/CTF70/CIX/9148/DEC
LCTC///L11//9148Y/AIR/00/PND
XPOS/302356Z10/DEC90/LL:835651N1-1733449E9
ENDAT
BT







NNNN
BT
MSGID/NPS/CIX/8310/NOV
LCTC///L11//8310W/OTHER/10/HOS
XPOS/282251Z99/NOV99/LL:751123N6-1655830E8
ENDAT
BT







NNNN
BT
MSGID/FTKNOX/CIX/5277/JUL
LCTC///L4A//5277N/SPACE/11/NEU
BMISL/YW9147883092//1/HOS////////1/211734Z65/JUL95
//LL:495832S2-1391830W4
ENDAT
BT







NNNN
BT
MSGID/TARDEC/CIX/7921/NOV
LCTC///L22//7921U/OTHER/01/UNK
XPOS/272149Z88/NOV98/LL:722529N1-1624438E5
ENDAT
BT
```

```
NNNN
BT
MSGID/COMTHIRDFLT/CIX/1159/FEB
LCTC///L11B//1159B/SUB/01/FRD
XPOS/111111Z21/FEB91/LL:123133S1-1023946W5
LEXT
ENDAT
BT
```

```
NNNN
BT
```

## 2. THE ABOVE MESSAGES IN XML FORMAT:

```
BT
MSGID/CTF70/CIX/9148/DEC
<CIX>
<LCTC>
      <LTYPE>L11/LTYPE>
      <LTN>9148Y/LTN>
      <LREPTYPE>AIR/LREPTYPE>
      <EXSIM>00/EXSIM>
      <THREAT>PND/THREAT>
</LCTC>
<XPOS>
      <DTG>302356Z10</DTG>
      <MOYR>DEC90</MOYR>
      <POSIT>LL:835651N1-1733449E9</POSIT>
</XPOS>
</CIX>
ENDAT
BT
```

```
NNNN
BT
MSGID/NPS/CIX/8310/NOV
<CIX>
<LCTC>
      <LTYPE>L11/LTYPE>
      <LTN>8310W/LTN>
      <LREPTYPE>OTHER/LREPTYPE>
      <EXSIM>10/EXSIM>
      <THREAT>HOS/THREAT>
</LCTC>
<XPOS>
      <DTG>282251Z99</DTG>
      <MOYR>NOV99</MOYR>
      <POSIT>LL:751123N6-1655830E8</POSIT>
</XPOS>
</CIX>
ENDAT
BT
```

```
NNNN
BT
MSGID/FTKNOX/CIX/5277/JUL
<CIX>
<LCTC>
      <LTYPE>L4A/LTYPE>
      <LTN>5277N/LTN>
      <LREPTYPE>SPACE/LREPTYPE>
      <EXSIM>11/EXSIM>
      <THREAT>NEU/THREAT>
</LCTC>
<BMISL>
      <UID>YW9147883092</UID>
      <TTYP>1</TTYP>
      <THREAT>HOS/THREAT>
      <RETYP>1</RETYP>
      <DTG>211734Z65</DTG>
      <MOYR>JUL95</MOYR>
      <POSIT>LL:495832S2-1391830W4</POSIT>
</BMISL>
</CIX>
ENDAT
BT
```

```
NNNN
BT
MSGID/TARDEC/CIX/7921/NOV
<CIX>
<LCTC>
      <LTYPE>L22/LTYPE>
      <LTN>7921U/LTN>
      <LREPTYPE>OTHER/LREPTYPE>
      <EXSIM>01/EXSIM>
      <THREAT>UNK/THREAT>
</LCTC>
<XPOS>
      <DTG>272149Z88</DTG>
      <MOYR>NOV98</MOYR>
      <POSIT>LL:722529N1-1624438E5</POSIT>
</XPOS>
</CIX>
ENDAT
BT
```

```
NNNN
BT
MSGID/COMTHIRDFLT/CIX/1159/FEB
<CIX>
<LCTC>
      <LTYPE>L11B/LTYPE>
      <LTN>1159B/LTN>
      <LREPTYPE>SUB/LREPTYPE>
      <EXSIM>01/EXSIM>
      <THREAT>FRD/THREAT>
</LCTC>
<XPOS>
      <DTG>111111Z21</DTG>
      <MOYR>FEB91</MOYR>
      <POSIT>LL:123133S1-1023946W5</POSIT>
</XPOS>
<LEXT>
</LEXT>
</CIX>
ENDAT
BT
```

```
NNNN
```

# APPENDIX M: SAMPLE RUN TIME LIST FOR MTF MESSAGE

## 1. MESSAGES IN NATIVE FORMAT:

```
BT
EXER/OLIVE_DRAB//
MSGID/NBC2/CTG810//
WEATHLOC/2318S11312W//
OBSTIME/141318Z//
WIND/142/-/25//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  / -25/    -20/   111/  14/
/M10KFT / -44/    -39/   163/  32//
BT
```

```
NNNN
BT
EXER/DESERT_WIND//
MSGID/NBC2/COMTHIRDFLT//
WEATHLOC/1423S10415W//
OBSTIME/111112Z//
WIND/123/-/18//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  / -13/     -8/   308/  82/
/M10KFT / -37/    -32/   320/  86//
BT
```

```
NNNN
BT
EXER/OLIVE_DRAB//
MSGID/NBC3/KITTYHAWK//
WEATHLOC/3050S12053W//
OBSTIME/161423Z//
WIND/135/-/22//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  /  71/     76/   233/  56/
/M10KFT /  78/     83/   134/  22//
BT
```

```
NNNN
BT
EXER/DESERT_WIND//
MSGID/NBC2/COMTHIRDFLT//
WEATHLOC/1717S10736W//
OBSTIME/121215Z//
WIND/122/-/18//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  / -30/    -25/   226/  54/
/M10KFT /  27/     32/   249/  62//
BT
```

```
NNNN
BT
EXER/MAHI_MAHI//
MSGID/TARGET/NPS//
WEATHLOC/7417N16415E//
OBSTIME/282250Z//
WIND/137/-/23//
5UPERAIR
/ALT    /TMPC/DEWTEMP/WINDIR/SPD
/M5KFT  / -30/    -25/   308/  82/
/M10KFT / -37/    -32/   172/  35//
BT
```

```
NNNN
BT
```

## 2. THE SAME MESSAGES IN XML FORMAT

```
BT
<exercise_identification>
  <exercise_nickname>OLIVE_DRAB</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>NBC2</message_text_format_identifier>
  <originator>CTG810</originator>
```

```
</message_identification>
<location_of_weather>
  <latitude_degrees>23</latitude_degrees>
  <latitude_minutes>18</latitude_minutes>
  <latitude_hemisphere>S</latitude_hemisphere>
  <longitude_degrees>113</longitude_degrees>
  <longitude_minutes>12</longitude_minutes>
  <longitude_hemisphere>W</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>14</day>
  <time_hour>13</time_hour>
  <time_min>18</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>
<wind>
  <wind_direction>142</wind_direction>
  <variable_wind_direction/>
  <wind_speed>25</wind_speed>
</wind>
<5uperair_data>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-25</temperature_celsius>
    <dewpoint_temperature_celsius>-20</dewpoint_temperature_celsius>
    <wind_direction>111</wind_direction>
    <peak_gusts_knots>14</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-44</temperature_celsius>
    <dewpoint_temperature_celsius>-39</dewpoint_temperature_celsius>
    <wind_direction>163</wind_direction>
    <peak_gusts_knots>32</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
BT




NNNN
BT
<exercise_identification>
  <exercise_nickname>DESERT_WIND</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>NBC2</message_text_format_identifier>
  <originator>COMTHIRDFLT</originator>
</message_identification>
<location_of_weather>
  <latitude_degrees>14</latitude_degrees>
  <latitude_minutes>23</latitude_minutes>
  <latitude_hemisphere>S</latitude_hemisphere>
  <longitude_degrees>104</longitude_degrees>
  <longitude_minutes>15</longitude_minutes>
  <longitude_hemisphere>W</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>11</day>
  <time_hour>11</time_hour>
  <time_min>12</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>
<wind>
  <wind_direction>123</wind_direction>
  <variable_wind_direction/>
  <wind_speed>18</wind_speed>
</wind>
<5uperair_data>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-13</temperature_celsius>
    <dewpoint_temperature_celsius> -8</dewpoint_temperature_celsius>
    <wind_direction>308</wind_direction>
    <peak_gusts_knots>82</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-37</temperature_celsius>
    <dewpoint_temperature_celsius>-32</dewpoint_temperature_celsius>
    <wind_direction>320</wind_direction>
    <peak_gusts_knots>86</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
BT




NNNN
BT
<exercise_identification>
  <exercise_nickname>OLIVE_DRAB</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>NBC3</message_text_format_identifier>
  <originator>KITTYHAWK</originator>
</message_identification>
<location_of_weather>
  <latitude_degrees>30</latitude_degrees>
  <latitude_minutes>50</latitude_minutes>
  <latitude_hemisphere>S</latitude_hemisphere>
  <longitude_degrees>120</longitude_degrees>
  <longitude_minutes>53</longitude_minutes>
  <longitude_hemisphere>W</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>16</day>
  <time_hour>14</time_hour>
  <time_min>23</time_min>
  <time_zone>Z</time_zone>
```

```
</observation_day_time>
<wind>
  <wind_direction>135</wind_direction>
  <variable_wind_direction/>
  <wind_speed>22</wind_speed>
</wind>
<5uperair_data>
  <5uperair_data_group_of_fields>
    <temperature_celsius> 71</temperature_celsius>
    <dewpoint_temperature_celsius> 76</dewpoint_temperature_celsius>
    <wind_direction>233</wind_direction>
    <peak_gusts_knots>56</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius> 78</temperature_celsius>
    <dewpoint_temperature_celsius> 83</dewpoint_temperature_celsius>
    <wind_direction>134</wind_direction>
    <peak_gusts_knots>22</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
BT
```

```
NNNN
BT
<exercise_identification>
  <exercise_nickname>DESERT_WIND</exercise_nickname>
</exercise_identification>
<message_identification>
  <message_text_format_identifier>NBC2</message_text_format_identifier>
  <originator>COMTHIRDFLT</originator>
</message_identification>
<location_of_weather>
  <latitude_degrees>17</latitude_degrees>
  <latitude_minutes>17</latitude_minutes>
  <latitude_hemisphere>S</latitude_hemisphere>
  <longitude_degrees>107</longitude_degrees>
  <longitude_minutes>36</longitude_minutes>
  <longitude_hemisphere>W</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>12</day>
  <time_hour>12</time_hour>
  <time_min>15</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>
<wind>
  <wind_direction>122</wind_direction>
  <variable_wind_direction/>
  <wind_speed>18</wind_speed>
</wind>
<5uperair_data>
```

```
  <5uperair_data_group_of_fields>
    <temperature_celsius>-30</temperature_celsius>
    <dewpoint_temperature_celsius>-25</dewpoint_temperature_celsius>
    <wind_direction>226</wind_direction>
    <peak_gusts_knots>54</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius> 27</temperature_celsius>
    <dewpoint_temperature_celsius> 32</dewpoint_temperature_celsius>
    <wind_direction>249</wind_direction>
    <peak_gusts_knots>62</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
BT
```

```
NNNN
BT
<exercise_identification>
  <exercise_nickname>MAHI_MAHI</exercise_nickname>
</exercise_identification>
<message_identification>

<message_text_format_identifier>TARGET</message_text_format_identifier>
  <originator>NPS</originator>
</message_identification>
<location_of_weather>
  <latitude_degrees>74</latitude_degrees>
  <latitude_minutes>17</latitude_minutes>
  <latitude_hemisphere>N</latitude_hemisphere>
  <longitude_degrees>164</longitude_degrees>
  <longitude_minutes>15</longitude_minutes>
  <longitude_hemisphere>E</longitude_hemisphere>
</location_of_weather>
<observation_day_time>
  <day>28</day>
  <time_hour>22</time_hour>
  <time_min>50</time_min>
  <time_zone>Z</time_zone>
</observation_day_time>
<wind>
  <wind_direction>137</wind_direction>
  <variable_wind_direction/>
  <wind_speed>23</wind_speed>
</wind>
<5uperair_data>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-30</temperature_celsius>
    <dewpoint_temperature_celsius>-25</dewpoint_temperature_celsius>
    <wind_direction>308</wind_direction>
    <peak_gusts_knots>82</peak_gusts_knots>
  </5uperair_data_group_of_fields>
  <5uperair_data_group_of_fields>
    <temperature_celsius>-37</temperature_celsius>
    <dewpoint_temperature_celsius>-32</dewpoint_temperature_celsius>
    <wind_direction>172</wind_direction>
    <peak_gusts_knots>35</peak_gusts_knots>
  </5uperair_data_group_of_fields>
<5uperair_data>
BT







NNNN
```

THIS PAGE INTENTIONALLY LEFT BLANK

## ACRONYMS

| | |
|---|---|
| ABCS | Army Battlefield Command Systems |
| AdaTP3 | Allied Data Publication, Number 3 |
| AFATDS | Advanced Field Artillery Tactical Data System |
| AIS | Automated Information System |
| API | Application Program Interfaces |
| C4I | Command, Control, Communications, Computers, and Intelligence |
| CIX | Coalition Information Exchange |
| COP | Common Operational Picture |
| DII COE | Defense Information Infrastructure Common Operating Environment |
| DoD | Department of Defense |
| GCCS | Global Command and Control System |
| HTML | HyperText Markup Language |
| I3 | Integrated Imagery and Intelligence |
| ISDS | Intelligence Shared Data Server |
| JBC | Joint Battle Center |
| JCDB | Joint Common Database |
| MTF | Message Text Format |
| MIDB | Modernized Integrated Database |
| NPS | Naval Postgraduate School |
| OS-OTG | Operational Specification for Over-the-Horizon Targeting Gold |
| OTH-G | Over The Horizon Targeting – Gold |
| SGML | Standard Generalized Markup Language |
| TADIL | Tactical Digital Information Links |
| TBM | Theater Ballistic Missile |
| TDBM | Track Database Manager |
| TMS | Track Management System |
| UB | Unified Build |
| UCP | Universal Comms Processor |
| USMTF | United States Message Text Format |
| VMF | Variable Message Format |
| WWW | World Wide Web |
| W3C | World Wide Web Consortium |
| XML | Extensible Mark-up Language |

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES

| | |
|---|---|
| BL90 | Berzins, V., and Luqi, *Software Engineering with Abstractions*, Addison-Wesley, Reading, MA, 1990 |
| IRI00 | International Research Institute Report, *Implementation Guidelines for Interoperability with the GCS-COP for JWID 2000*, May 2000 |
| MC98 | Malloy and Schneider of Mitre Corp., *Experiences Designing Query Languages for Hierarchically Structured Text Documents*, [http://www.w3.org/TandS/QL/QL98/pp/mitre.html], September 1998 |
| MIL1 | Department of Defense Military Specification MIL-STD-6040; USMTF Message Formatting Program |
| MTF00 | USMTF User Formats Baseline Electronic Documentation System, DISA/JIEO, *www-usmtf.itsi.disa.mil*, Feb 2000 |
| MTFWS | USMTF Workbook Supplement, JID FORSCOM, September 1999 |
| RH99 | Hina, D., *Evaluation of the Extensible Markup Language (XML) as a means for Establishing Interoperability between Heterogeneous Department of Defense (DoD) Databases*, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2000 |
| RS96 | Renner, S., and Scarano, J., "Migrating Legacy Applications to a Shared Data Environment", *Proceedings of the Federal Database Colloguium and Exposition*, August 1996 |
| XML1 | W3C Report, *XML-MTF Mapping Third Public Working Draft*, by Mike Cokus of Mitre Corp., August, 2000 |
| XML2 | Idris, *Benefits of Using XML*, [65.1.136.127/developerlife/xmlbenefits/default.htm], June 1999 |
| XML3 | Naval Center for Tactical Systems Interoperability Report, *eXtensible Markup Language (XML) – A Tool for Interoperability*, by Brian Hopkins, June 1999 |
| YBG02 | Young, P., Berzins, V. Ge, J., Luqi, "Using an Object Oriented Model for Resolving Representational Differences between Heterogeneous Systems", *The 17th ACM Symposium on Applied Computing*, Madrid, Spain, March 10-14, 2002. |

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Engineering & Technology Curriculum
   Code 34
   Naval Postgraduate School
   Monterey, California

4. Computer and Information Programs Office
   Code 32
   Naval Postgraduate School
   Monterey, California

5. Dr Luqi
   Naval Postgraduate School
   Monterey, California

5. Dr Valdis Berzins
   Naval Postgraduate School
   Monterey, California

6. Captain Paul E. Young
   305 Morning Dove Way
   Arnold, Maryland 21012

7. Kris Pradeep
   AMSTA-TR-R/265,
   US Army TACOM,
   Warren, Michigan